
How Reinforcement Learning After Next-Token Prediction Facilitates Learning

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 To study the success of the current post-training recipe of Large Language Models,
2 we consider a setting of learning mixture distributions consisting of short and long
3 “chain-of-thought” sequences encoding the parity of d bits. When long sequences
4 are rare, we show how reinforcement learning after next-token prediction enables
5 autoregressive transformers to generalize, whereas mere next-token prediction
6 requires extreme statistical or computational resources to do so. We further explain
7 how reinforcement learning leverages increased test-time computation, manifested
8 in longer responses, to facilitate this learning process. In a simplified setting, we
9 prove that autoregressive linear models following this training recipe can efficiently
10 learn to predict the parity of d bits as long as the proportion of long demonstrations
11 in the data mix is not exponentially small in the input dimension d . Finally, we
12 demonstrate the same phenomena in less idealized settings with Llama models.

13 1 Introduction

14 The application of reinforcement learning techniques to large neural networks has led to many
15 machine learning advances and successes [38, 39, 5, 3]. In the realm of Large Language Models
16 (LLMs), a recent paradigm that leverages reinforcement learning consists of two main ingredients:
17 training a large model, such as an autoregressive transformer [42] on diverse types of data (sequences)
18 from the web with a next-token prediction objective [30], followed by a fine-tuning stage with a
19 reinforcement learning algorithm that seeks to improve model generations with respect to a reward
20 function. When the latter part is applied on a frontier model using a mathematical, logical or
21 reasoning reward, it leads to rapid increase in the model’s capabilities in related domains and is
22 often accompanied by a significant length increase in model response. Prominent examples of such
23 “reasoning” language models include OpenAI’s o1 model [13] and Deepseek’s R1 model [7]. The
24 prolonged LLM generation that precedes an answer has been dubbed the “thinking process” of the
25 model, as it often resembles the “chain-of-thought” [28, 44] of a human expert solving said task.

26 In this paper, we study how autoregressive models succeed in solving challenging prediction tasks
27 by following this training recipe (next-token prediction followed by reinforcement learning¹). To
28 provide a theoretical account, we make a central simplification: we assume that pre-training data
29 already contain correct and elaborate but perhaps *rare* demonstrations for a task of interest. Based on
30 this modeling assumption, we study and explain:

- 31 1. Why in certain cases it is difficult for a model to generalize during pre-training.
- 32 2. How reinforcement learning (RL) leads to a rapid improvement in terms of samples, and
- 33 3. What optimization pressures cause increase in the length of the response.

¹In the argot of LLMs, these two stages are often called pre-training and post-training, respectively.

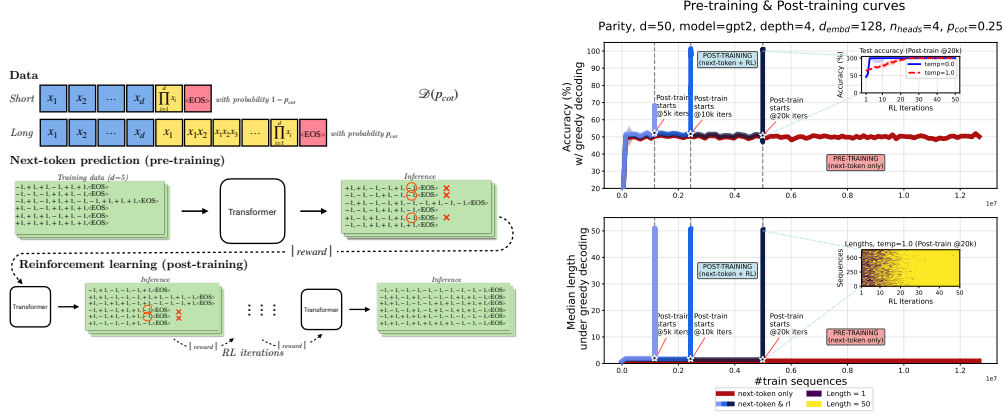


Figure 1: *Left: An illustration of our main learning setting:* a mixture of long and short sequences encoding the parity of d bits, along with a representation of pre- and post-training for $d=5$. *Right: Advantage of next-token prediction followed by reinforcement learning over mere next-token prediction* in predicting the parity of 50 bits with a transformer trained from scratch. The red line corresponds to pre-training, while blue lines correspond to the same pre-training runs, each followed by GRPO (with a final token accuracy reward) from a different checkpoint. Lines correspond to median across 3 seeds. *Top:* Test accuracy under greedy decoding. *Bottom:* Median length of model response under greedy decoding. The inset figures zoom-in on the curves during post-training.

Specifically, we study learning from a distribution that contains both short and long, chain-of-thought-like, sequences encoding a single task. The main body of the paper is devoted to the task of predicting the parity of d bits [26, 17, 34, 4], while we consider more variations of this learning setup in the Appendix. Through extensive empirical simulations on transformers trained from scratch, we show that if the task complexity is high (large input dimension d) and long demonstrations are scarce, then mere next-token prediction results in a model that fails to predict the parity on unseen examples. On the other hand, when switching to an RL algorithm with a correctness reward, model performance improves rapidly and the length of the generated sequences increases. The results hold for a large array of model choices, problem and optimization hyperparameters, and reinforcement learning algorithms. Figure 1 (Left: Illustration, Right: Simulations) summarizes the landscape.

In Section 3, we theoretically capture and explain most of the previously observed phenomena in a simplified setting. Let p_{cot} be the proportion of long data in the parity mixture distribution $\mathcal{D}(p_{\text{cot}})$. We analyze pre-training (next-token prediction) with Stochastic Gradient Descent on $\mathcal{D}(p_{\text{cot}})$ for an autoregressive architecture that consists of a series of linear models [22], while for post-training, we consider the STaR algorithm [49] (which is a REINFORCE [46]-type algorithm) with a reward that verifies correctness of the whole chain-of-thought. In this setting, we prove that the model: (a) fails to generalize under greedy decoding during the course of pre-training if long demonstrations are rare (i.e., if $p_{\text{cot}} < 1/3$, which matches the empirical threshold observed with transformers) (Theorem 1), (b) learns fast from long demonstrations and remains length-calibrated (Theorem 1), and (c) succeeds in generalizing after the completion of both pre- and post-training in $O(\text{poly}(d))$ SGD iterations, provided that long demonstrations are not exponentially rare in the data distribution, while this is accompanied with length increase during RL (Theorem 2). The analysis captures the immediate progress seen in practice: as we show, only $O\left(\log \frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)$ RL rounds suffice to obtain a generalizing model. To the best of our knowledge, this suite of results provides the first theoretical separation between next-token prediction and next-token prediction combined with RL in the autoregressive setting, as well as the first optimization result demonstrating length increase during RL in LLMs.

2 Experiments on Parity

Setup We study the task of predicting the parity of d bits given access to a source of sequences $\mathcal{D}(p_{\text{cot}})$ which either consist of: (i) input bits and their parity, or (ii) input bits, intermediate computations and the final parity (Figure 1). We train decoder-only autoregressive, transformers [42]. In

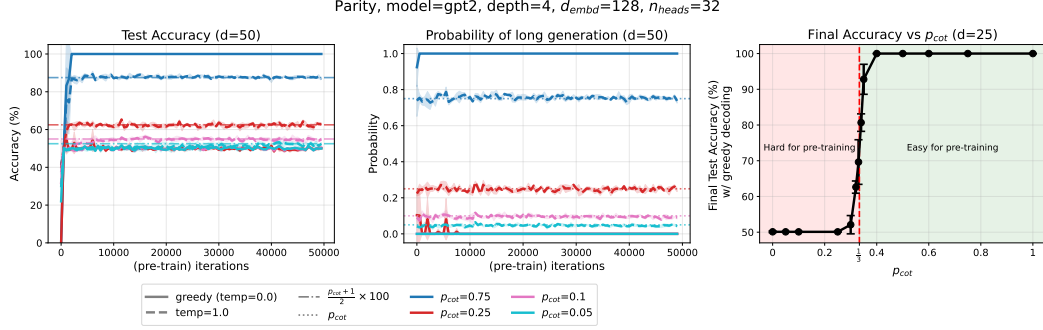


Figure 2: **Pre-training of transformers on mixture of long and short sequences encoding the parity of d bits.** *Left:* Test accuracy during the course of pre-training. *Center:* The probability that a model’s generation length is equal to the maximum length present in the training data (which equals d). *Right:* Test accuracy with greedy decoding at the end of pre-training vs mixture coefficient p_{cot} .

the first stage of training (pre-training), we train a GPT2-type model to estimate the probability of the next token on sequences drawn from $\mathcal{D}(p_{\text{cot}})$ [30]. After several steps of pre-training, we switch training algorithm and further optimize the model with reinforcement learning methods; *STaR* [49], *REINFORCE* [46] and *GRPO* [35]. At a high level, all three algorithms seek to maximize the reward of model generations (see Appendix B for details). We focus on two types of rewards:

End-to-end correctness: The reward function assesses whether the last token is equal to the correct answer: $r_{\text{e2e}}(\mathbf{x}, y) = \mathbb{1} \left\{ y[-1] = \prod_{i=1}^d x_i \right\}$, where $y[-1] \in \{\pm 1\}$ denotes the token appearing right before the $\langle \text{EOS} \rangle$ token in sequence y .

Chain-of-thought correctness: The reward function assesses whether the whole sequence is valid: $r_{\text{cot}}(\mathbf{x}, y) = \mathbb{1} \left\{ y = \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \vee y = \left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \right\}$.

Results First, we show results for pre-training (next-token prediction) on data sampled from $\mathcal{D}(p_{\text{cot}})$ for various values of mixture weight p_{cot} in Figure 2. As we can see, performance under greedy decoding is determined by the value of p_{cot} : if the percentage of long sequences in the data is large enough (e.g. $p_{\text{cot}}=0.75$), then the model quickly learns to predict the parity of the input correctly by generating long responses and generalizes perfectly. On the other hand, when long sequences are rare, the model generates short responses and performs on par with random guessing even after training on millions of sequences. In fact, the critical threshold is around $p_{\text{cot}}=1/3$ (Figure 2 (right)).

For the distributions with small p_{cot} where the models did not manage to generalize during pre-training, we consider switching to post-training: at some iteration, we stop training with the next-token objective and resume model training with an RL algorithm. We show results in Figure 3 for $p_{\text{cot}}=0.25$, where we plot post-training accuracy as a function of RL iterations for all post-train algorithms. Progress during RL is immediate: after training on only a few sequences, model performance under greedy decoding grows from random guessing to 100% accuracy. This improvement is accompanied with increase in the length of the model’s generations, which grows from 1 (min train length) to 50 (max train length). To further emphasize the sample complexity difference between the two paradigms, we show aggregated results (pre- & post-training combined) in Figure 1 for $d=50$, $p_{\text{cot}}=0.25$ and post-training with GRPO starting from various checkpoints. Appendix D.1 contains more discussion and results. Next, we theoretically elucidate the mechanisms by which RL improves over next-token prediction in our setting and why this improvement is accompanied by length increase.

3 Theory

Architecture We study an autoregressive model that consists of $d+1$ linear predictors. We reduce next-token prediction on $\mathcal{D}(p_{\text{cot}})$ to a series of binary classification problems. At a high-level, the feature embedding used in each one of them involves (at most) second-degree monomials of the input, which is a canonical embedding choice. We denote the induced autoregressive hypothesis class by $\mathcal{H}_{\text{AR}}^{\text{Lin}}$. We define GREEDY to be the operation of greedy decoding from the model logits (takes the sign of each model’s output). By composing each individual component of $h \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ with the sigmoid,

we also define the probability measure induced by h , namely $\pi_h(\cdot \mid \mathbf{x}) \in \Delta(\mathcal{Y})$. This corresponds to sampling from the model with temperature 1. Appendix F contains background information, formal definitions and proofs of all statements. Our first result characterizes the model during pre-training.

Theorem 1. (*Pre-training, Informal*) Let $d \geq 2$, $p_{\text{cot}} \in (0, 3/4)$. Consider running Stochastic Gradient Descent (SGD) for minimizing the next-token prediction objective with respect to distribution $\mathcal{D}(p_{\text{cot}})$ over $\mathcal{H}_{\text{AR}}^{\text{Lin}}$ with an additional ℓ_2 regularization term. Then, for any $0 < \varepsilon < (d+1) \min \left\{ \frac{1}{2}, \ln \frac{1+p_{\text{cot}}}{1-p_{\text{cot}}}, \left\lfloor \ln \frac{1-p_{\text{cot}}}{2p_{\text{cot}}} \right\rfloor \right\}$ and after $O(\text{poly}(d, 1/\varepsilon, 1/p_{\text{cot}}))$ iterations, with high probability over sampling from $\mathcal{D}(p_{\text{cot}})$, SGD returns model $h_{\text{pre}} \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ for which the following hold:

1. **Length calibration:** For any input $\mathbf{x} \in \{\pm 1\}^d$, it holds:

$$\begin{aligned} \left| \pi_{h_{\text{pre}}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \mid \mathbf{x} \right) - p_{\text{cot}} \right| &\lesssim \varepsilon, \\ \left| \pi_{h_{\text{pre}}} \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \mid \mathbf{x} \right) - \frac{1-p_{\text{cot}}}{2} \right| &\lesssim \varepsilon. \end{aligned} \quad (1)$$

2. **Failure of greedy decoding:** For any $\mathbf{x} \in \{\pm 1\}^d$, it holds:

$$\text{GREEDY}(h_{\text{pre}}(\mathbf{x})) = \begin{cases} (x_1, \langle \text{EOS} \rangle), & \text{if } p_{\text{cot}} < 1/3, \\ (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle), & \text{otherwise.} \end{cases} \quad (2)$$

That is, test accuracy under greedy decoding is on par with random chance if $p_{\text{cot}} < 1/3$, and perfect otherwise.

Next, we study RL with (regularized) STaR objective and show the main learning result of the paper.

Theorem 2. (*Post-training, Informal*) Let h_{pre} be the output of SGD after pre-training under the conditions of Theorem 1 for $p_{\text{cot}} < 1/3$ and $\varepsilon \leq c_0 \frac{p_{\text{cot}}}{1-p_{\text{cot}}}$ for a sufficiently small constant $c_0 > 0$. Consider post-training of h_{pre} with the STaR algorithm using reward r_{cot} . Suppose each STaR round uses $O(\text{poly}(d, 1/\varepsilon, 1/p_{\text{cot}}))$ SGD iterations, and let $h^{(n)}$ denote the model after n rounds. Then, there exists an integer $n^* = O\left(\log \frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)$ such that with high probability over sampling from $\mathcal{D}(p_{\text{cot}})$, and the model-sampled outputs used by STaR in rounds $1, \dots, n^* - 1$, the following hold:

1. **Length increases:** The probability of a long generation increases:

$$\left| \pi_{h^{(n)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \mid \mathbf{x} \right) - q_n \right| \lesssim \varepsilon, \quad (3)$$

where $|q_n - p_n| \lesssim 2^n \varepsilon$ and $p_{n+1} = \frac{2p_n}{1+p_n}$ for all $n \leq n^*$ with $p_0 = p_{\text{cot}}$.

2. **Perfect generalization:** After n^* RL rounds, it holds:

$$\text{GREEDY}(h^{(n^*)}(\mathbf{x})) = \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right). \quad (4)$$

Remark 1. Consider the case where long demonstrations are very rare, that is $p_{\text{cot}} \rightarrow 0$ as $d \rightarrow \infty$, which is perhaps the most interesting case. Theorem 2 shows that as long as $p_{\text{cot}} \in \Omega(d^{-\kappa})$ for some constant $\kappa \in \mathbb{N}$, then we can learn the parity using $O(\text{poly}(d))$ SGD steps. This should be contrasted with the well-known hardness results on learning parities [17, 37].

4 Conclusion

In this work, we introduced a theoretical framework to study the success of reinforcement learning applied after next-token prediction in Large Language Models. We demonstrated and proved that when the data contain rare elaborate sequences encoding a challenging target function, RL can help the model to learn by effectively up-sampling the presence of the long demonstrations in the data mix. Future work can address the limitations of our setting, by understanding, for example, how noisy chains of thought affect the conclusions, as well as considering separate pre-train and post-train target functions. Experiments with LLMs can be found in Appendix E and a discussion in Appendix A.

References

- [1] Abbe, E., Adserà, E. B., and Misiakiewicz, T. (2023). SGD learning on neural networks: leap complexity and saddle-to-saddle dynamics. In Neu, G. and Rosasco, L., editors, *The Thirty Sixth Annual Conference on Learning Theory, COLT 2023, 12-15 July 2023, Bangalore, India*, volume 195 of *Proceedings of Machine Learning Research*, pages 2552–2623. PMLR.
- [2] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Al-tenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- [3] Bakhtin, A., Wu, D. J., Lerer, A., Gray, J., Jacob, A. P., Farina, G., Miller, A. H., and Brown, N. (2023). Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
- [4] Barak, B., Edelman, B. L., Goel, S., Kakade, S. M., Malach, E., and Zhang, C. (2022). Hidden progress in deep learning: SGD learns parities near the computational limit. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- [5] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680.
- [6] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.
- [7] DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., and Li, S. S. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.
- [8] Deng, Y., Choi, Y., and Shieber, S. M. (2024). From explicit cot to implicit cot: Learning to internalize cot step by step. *CoRR*, abs/2405.14838.
- [9] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. (2024). The llama 3 herd of models. *CoRR*, abs/2407.21783.

- [10] Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. (2023). Towards revealing the mystery behind chain of thought: A theoretical perspective. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- [11] Hastad, J. (1987). *Computational Limitations of Small-Depth Circuits*. MIT Press (MA).
- [12] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- [13] Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., Iftimie, A., Karpenko, A., Passos, A. T., Neitz, A., Prokofiev, A., Wei, A., Tam, A., Bennett, A., Kumar, A., Saraiva, A., Vallone, A., Duberstein, A., Kondrich, A., Mishchenko, A., Applebaum, A., Jiang, A., Nair, A., Zoph, B., Ghorbani, B., Rossen, B., Sokolowsky, B., Barak, B., McGrew, B., Minaiev, B., Hao, B., Baker, B., Houghton, B., McKinzie, B., Eastman, B., Lugaresi, C., Bassin, C., Hudson, C., Li, C. M., de Bourcy, C., Voss, C., Shen, C., Zhang, C., Koch, C., Orsinger, C., Hesse, C., Fischer, C., Chan, C., Roberts, D., Kappler, D., Levy, D., Selsam, D., Dohan, D., Farhi, D., Mely, D., Robinson, D., Tsipras, D., Li, D., Oprea, D., Freeman, E., Zhang, E., Wong, E., Proehl, E., Cheung, E., Mitchell, E., Wallace, E., Ritter, E., Mays, E., Wang, F., Such, F. P., Raso, F., Leoni, F., Tsimpourlas, F., Song, F., von Lohmann, F., Sulit, F., Salmon, G., Parascandolo, G., Chabot, G., Zhao, G., Brockman, G., Leclerc, G., Salman, H., Bao, H., Sheng, H., Andrin, H., Bagherinezhad, H., Ren, H., Lightman, H., Chung, H. W., Kivlichan, I., O’Connell, I., Osband, I., Gilaberte, I. C., and Akkaya, I. (2024). Openai o1 system card. *CoRR*, abs/2412.16720.
- [14] Jelassi, S., Brandfonbrener, D., Kakade, S. M., and Malach, E. (2024). Repeat after me: Transformers are better than state space models at copying. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.
- [15] Joshi, N., Vardi, G., Block, A., Goel, S., Li, Z., Misiakiewicz, T., and Srebro, N. (2025). A theory of learning with autoregressive chain of thought. In Haghtalab, N. and Moitra, A., editors, *The Thirty Eighth Annual Conference on Learning Theory, 30-4 July 2025, Lyon, France*, volume 291 of *Proceedings of Machine Learning Research*, pages 3161–3212. PMLR.
- [16] Kamath, P., Montasser, O., and Srebro, N. (2020). Approximate is good enough: Probabilistic variants of dimensional and margin complexity. In *Conference on Learning Theory*, pages 2236–2262. PMLR.
- [17] Kearns, M. J. (1998). Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006.
- [18] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- [19] Li, Z., Liu, H., Zhou, D., and Ma, T. (2024). Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
- [20] Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2023). Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
- [21] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.
- [22] Malach, E. (2024). Auto-regressive next-token predictors are universal learners. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.
- [23] Malach, E. and Shalev-Shwartz, S. (2022). When hardness of approximation meets hardness of learning. *J. Mach. Learn. Res.*, 23:91:1–91:24.

- [24] Merrill, W. and Sabharwal, A. (2023). The parallelism tradeoff: Limitations of log-precision transformers. *Trans. Assoc. Comput. Linguistics*, 11:531–545.
- [25] Merrill, W. and Sabharwal, A. (2024). The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
- [26] Minsky, M. and Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press.
- [27] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press.
- [28] Nye, M. I., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. (2021). Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.
- [29] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- [30] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI*.
- [31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.
- [32] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In Erk, K. and Smith, N. A., editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany. Association for Computational Linguistics.
- [33] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.
- [34] Shalev-Shwartz, S., Shamir, O., and Shammah, S. (2017). Failures of gradient-based deep learning. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3067–3075. PMLR.
- [35] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.
- [36] Shen, R., Bubeck, S., Eldan, R., Lee, Y. T., Li, Y., and Zhang, Y. (2023). Positional description matters for transformers arithmetic. *CoRR*, abs/2311.14737.
- [37] Shoshani, I. and Shamir, O. (2025). Hardness of learning fixed parities with neural networks. *CoRR*, abs/2501.00817.
- [38] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.
- [39] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

- 283 [40] Su, J., Ahmed, M. H. M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). Roformer: Enhanced
284 transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- 285 [41] torchtune maintainers and contributors (2024). torchtune: Pytorch’s finetuning library.
- 286 [42] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L.,
287 and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio,
288 S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in*
289 *Neural Information Processing Systems 30: Annual Conference on Neural Information Processing*
290 *Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.
- 291 [43] von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang,
292 S., Rasul, K., and Gallouédec, Q. (2020). Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>.
- 293
- 294 [44] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and
295 Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In
296 Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in*
297 *Neural Information Processing Systems 35: Annual Conference on Neural Information Processing*
298 *Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- 299 [45] Wies, N., Levine, Y., and Shashua, A. (2023). Sub-task decomposition enables learning in
300 sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations,*
301 *ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
- 302 [46] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist
303 reinforcement learning. *Machine Learning*, 8:229–256.
- 304 [47] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf,
305 R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C.,
306 Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-
307 the-art natural language processing. In Liu, Q. and Schlangen, D., editors, *Proceedings of the*
308 *2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*,
309 pages 38–45, Online. Association for Computational Linguistics.
- 310 [48] Wu, J., Bartlett, P. L., Telgarsky, M., and Yu, B. (2025). Benefits of early stopping in gradient
311 descent for overparameterized logistic regression. *CoRR*, abs/2502.13283.
- 312 [49] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. (2022). Star: Bootstrapping reasoning
313 with reasoning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A.,
314 editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural*
315 *Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 -*
316 *December 9, 2022*.

317 Table of Contents

318	A Discussion	9
319	B Models and Algorithms	10
320	B.1 Architecture	10
321	B.2 Pre-training	10
322	B.3 Post-training	10
323	C Experimental Details	12
324	C.1 Parity	12
325	C.2 Number Multiplication	12
326	C.3 GSM8K	13
327	C.4 MATH	14
328	D Additional experimental results	15
329	D.1 More experiments on main parity setting	15
330	D.2 Partial chain of thought	22
331	E Experiments with Mathematical Reasoning Benchmarks	23
332	E.1 Number Multiplication	24
333	E.2 Grade School & High School Mathematics (GSM8K & MATH datasets)	25
334	F Proofs	44
335	F.1 Preliminaries & Setup	44
336	F.2 Pre-training	50
337	F.3 Post-training	64

338 A Discussion

339 Here, we discuss our results and their implications in more detail.

340 **Learning with chain of thought data** Recently, a few theoretical works have attempted to capture
341 the success of autoregressive modeling with LLMs [45, 22, 15]. In particular, Joshi et al. [15], Malach
342 [22] showed that next-token prediction can lead to computationally efficient learning of any efficiently
343 computable binary function, which stands in sharp contrast to standard supervised learning where only
344 a limited class of functions can be learned efficiently. However, these results rely on the assumption
345 that a learner has access to a dataset with perfect chain of thought data. While this has turned out to
346 be a very productive assumption for the theoretical study of autoregressive learning, it nonetheless
347 constitutes a strong assumption. Our work takes a step forward in relaxing this requirement; it
348 instead assumes that the dataset contains at least a nonzero (possibly polynomially small in the
349 context length) fraction of chain of thought data. This is arguably a more natural assumption for
350 modeling the presence of elaborate “good” demonstrations in the vast ocean of internet text. As
351 we showed in Section 3, this is enough to guarantee efficient learning of the parity function and,
352 interestingly, the guarantee is achieved through a variation of the popular post-training recipe used
353 in state-of-the-art LLMs. We believe our proof strategy can be modified to show that pre-training
354 followed by post-training can lead to efficient learning for other “hard” functions. We leave such a
355 study for future work.

Length increase as a learning phenomenon The common wisdom in the literature has been that the chain of thought during RL grows to enable the approximation of complex algorithmic tasks. Indeed, some computational problems require large computational depth to be executed with reasonable resources [11] and the standard decoder-only transformer architecture with context length n appears limited in what it can represent exactly with constant depth. On the other hand, a transformer augmented with $O(\text{poly}(n))$ chain of thought can simulate any function described by a circuit of polynomial size [10, 19, 24], as the additional chain of thought provides computational depth to the model. As such, it seems theoretically satisfying that when chain of thought grows during RL, the model’s performance improves on complex tasks. One shortcoming of this perspective is that it does not aim to explain why or how optimization pressures lead to length increase, which was the main focus of our work. On a more abstract level, the approximation advantage of long responses suggested by prior work is indeed fundamental, as one cannot hope to learn a task without the capacity to represent it². Nevertheless, the learning advantage we captured in this paper can be more prevalent as it even appears in cases where representation is not an issue (such as the parity task we considered). Based on the above, we suggest interpreting the growth of the length during reinforcement learning of autoregressive models as a bona fide learning phenomenon.

B Models and Algorithms

Herein, we describe the main models and training algorithms we consider in the parity experiments.

B.1 Architecture

A transformer is a sequence-to-sequence neural network, which in its simplest form consists of layers of self-attention followed by a position-wise feed-forward network. The term “autoregressive” indicates that the prediction of the model for each new token is conditioned only on tokens that appeared earlier in the sequence. To obtain a distribution over the next token, it is common to compose the output of the transformer with a soft-max layer across the vocabulary index, which can be potentially parameterized by a temperature value.

B.2 Pre-training

The next-token prediction objective consists of the log loss of the model token distribution evaluated at output positions 1 to d . We omit the first d input positions of the sequence from the loss objective, as input bits x_1, \dots, x_d are distributed uniformly at random. If we denote the predicted sequence-level distribution by $\pi_\vartheta(y \mid \mathbf{x}) = \prod_{j=1}^{|y|} \pi_{\vartheta,j}(y_j \mid \mathbf{x}, y_{<j}) \in (0, 1)$, which is induced by the prediction of the transformer at each position composed with the soft-max, then training corresponds to:

$$\min_{\vartheta} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}(p_{\text{cot}})} \left[\sum_{j=1}^{|y|} -\ln \pi_{\vartheta,j}(y_j \mid \mathbf{x}, y_{<j}) \right] \quad (5)$$

B.3 Post-training

We consider three main reinforcement learning algorithms: *Self-Taught Reasoner* (STaR) [49], *REward Increment=Nonnegative Factor times Offset Reinforcement times Characteristic Eligibility* (REINFORCE) [46] and *Group Relative Policy Optimization* (GRPO) [35]. Let $r : \mathcal{X} \times \mathcal{Y} \mapsto \{0, 1\}$ be a binary reward function. We consider outcome-based rewards, which assign one reward value per sequence.

STaR In STaR (Algorithm 1), we first query the pre-trained model to generate responses for inputs. We then filter out those with 0 reward and fine-tune the model on the rest using the next-token prediction objective. We repeat this process for several rounds, each time sampling from the model

²The actual situation is a bit more nuanced, as exact representation is not a strict requirement for learning through approximation (which is the goal in e.g. PAC learning). See [16, 23] for some definitions of approximation which can be more fruitful for understanding learning.

396 returned by the previous round. The algorithm over one round corresponds to:

$$\min_{\vartheta} \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta_{\text{prev}}}(\cdot | \mathbf{x})} \left[\sum_{j=1}^{|y|} -\ln \pi_{\vartheta, j}(y_j | \mathbf{x}, y_{< j}) \middle| r(\mathbf{x}, y) = 1 \right], \quad (6)$$

397 where $\vartheta_{\text{prev}} \in \mathbb{R}^p$ corresponds to parameters returned in the previous round of the algorithm.

398 The pseudocode of STaR can be found in Algorithm 1.

Algorithm 1 Self-Taught Reasoner (STaR) Algorithm

Require: Pre-trained model parameters ϑ_0 , RL rounds n , Fine-tuning epochs per round E , Input distribution $\mathcal{D}_{\mathbf{x}} = \text{Rad}(1/2)^{\otimes d}$, Reward function $r(\mathbf{x}, y)$, Number of samples to generate per round N .

```

1: Set  $\vartheta = \vartheta_0$ 
2: for  $r = 1$  to  $n$  do
3:   Set  $\mathcal{S} = \emptyset$ 
4:   for  $i = 1$  to  $N$  do
5:     Sample  $\mathbf{x} \sim \mathcal{D}_{\mathbf{x}}$ .
6:     Sample  $y \sim \pi_{\vartheta}(\cdot | \mathbf{x})$ .
7:     if  $r(\mathbf{x}, y) = 1$  then
8:       Set  $\mathcal{S} = \mathcal{S} \cup \{(\mathbf{x}, y)\}$ .
9:     end if
10:  end for
11:  for epoch = 1 to  $E$  do
12:    for each  $(\mathbf{x}, y) \in \mathcal{S}$  do
13:      Update  $\vartheta$  by taking a gradient step on the next-token prediction loss for  $(\mathbf{x}, y)$ .
14:    end for
15:  end for
16: end for
17: return Final model parameters  $\vartheta$ .

```

399 **REINFORCE** REINFORCE is a standard policy gradient algorithm. It seeks to maximize the
400 expected reward of a sequence generated by the model:

$$\max_{\vartheta} \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta}(\cdot | \mathbf{x})} [r(\mathbf{x}, y)]. \quad (7)$$

401 The gradient of the objective function can be written as:

$$\mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta}(\cdot | \mathbf{x})} [\nabla_{\vartheta} \ln \pi_{\vartheta}(y | \mathbf{x}) r(\mathbf{x}, y)], \quad (8)$$

402 which implies that the algorithm can be recast as:

$$\min_{\vartheta} \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta}(\cdot | \mathbf{x})} [-\ln \pi_{\vartheta}(y | \mathbf{x}) r(\mathbf{x}, y)], \quad (9)$$

403 where the random variable y is treated as a constant and not as a function of ϑ . This is the surrogate
404 objective that is optimized in practice. Observe the similarities with the STaR objective of (6).
405 Furthermore, a widely adopted option is to center the rewards in order to reduce the variance of the
406 updates during the execution of a stochastic optimization algorithm. This leads to the final version of
407 the REINFORCE method:

$$\min_{\vartheta} \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta}(\cdot | \mathbf{x})} \left[\sum_{j=1}^{|y|} -\ln \pi_{\vartheta, j}(y_j | \mathbf{x}, y_{< j}) A(\mathbf{x}, y) \right], \quad (10)$$

408 where $A(\mathbf{x}, y) = r(\mathbf{x}, y) - \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{\vartheta}(\cdot | \mathbf{x})} [r(\mathbf{x}, y)]$ is often called the advantage function.

409 **GRPO** Group Relative Policy Optimization, or GRPO for short, is a policy gradient algorithm,
 410 motivated by the class of Proximal Policy Optimization (PPO) algorithms [31], that is widely used
 411 for post-training large language models with reinforcement learning. Its objective amounts to:

$$\mathbb{E}_{\substack{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d} \\ y^{(1)}, \dots, y^{(N)} \sim \pi_{\text{old}}(\cdot | \mathbf{x})}} \left[\frac{1}{N} \sum_{i=1}^N \frac{1}{|y^{(i)}|} \sum_{j=1}^{|y^{(i)}|} \min\{r_{i,j} A_i, \text{clip}(r_{i,j}, 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}) A_i\} \right],$$

412 where $A_i = \frac{r(\mathbf{x}, y^{(i)}) - \text{mean}(r(\mathbf{x}, y^{(1)}), \dots, r(\mathbf{x}, y^{(N)}))}{\text{std}(r(\mathbf{x}, y^{(1)}), \dots, r(\mathbf{x}, y^{(N)}))}$ is a normalized reward, ϵ_{clip} is a hyperparameter
 413 and π_{old} is a predicted distribution that corresponds to an earlier model checkpoint during the
 414 execution of the algorithm. We defined $r_{i,j} := \frac{\pi_{\theta,j}(y_j^{(i)} | \mathbf{x}, y_{<j}^{(i)})}{\pi_{\text{old},j}(y_j^{(i)} | \mathbf{x}, y_{<j}^{(i)})}$. The term ‘‘Group’’ indicates
 415 sampling of a group of N responses per input.

416 C Experimental Details

417 The experiments are implemented using PyTorch [29].

418 C.1 Parity

419 We use the Transformers library [47] for our parity experiments.

420 **Architecture** The transformer is initialized with context length $d'=2d$, to be able to process
 421 the whole sequence, where recall d is the number of input bits. The two architectures we con-
 422 sider primarily differ in the type of positional encodings: GPT2 uses absolute positional encod-
 423 ings of the input sequence, while Mistral uses relative positional encodings [40]. We use the
 424 GPT2Config, MistralConfig config classes to define the models. We consider several model
 425 hyperparameters: depth (number of transformer blocks) $L \in \{2, 4, 8\}$, embedding dimension
 426 $d_{\text{emb}} \in \{128, 256\}$, number of heads $n_{\text{heads}} \in \{4, 32\}$. By the default convention, each head
 427 has dimension $d_{\text{emb}}/n_{\text{heads}}$.

Pre-training We consider distributions with

$$p_{\text{cot}} \in \{0, 0.05, 0.1, 0.25, 0.3, 0.32, 0.33, 0.34, 0.35, 0.4, 0.5, 0.6, 0.75, 1.0\}.$$

428 We use the Adam [18] optimizer with learning rate $\eta_{\text{pre}}=10^{-3}$ and a batch size $B_{\text{pre}}=256$. Test
 429 statistics are estimated using 2,560 new samples.

430 In Figure 2 (right), each bullet corresponds to the final test accuracy with greedy decoding over 3
 431 random seeds. The final accuracy of each run is defined as the average accuracy from the last 5
 432 checkpoints of the pre-trained run (that is, average accuracy at iterations 47,500 to 49,500).

433 **Post-training** We continue post-training from the last checkpoint of pre-training. For STaR, we
 434 generate 3, 200 samples per RL iteration and train on the correct ones for 3 epochs with batch size 64.
 435 For REINFORCE and GRPO, we use batch size 64. We found it necessary to use a larger sample size
 436 per RL iteration for STaR. The GRPO experiments use $N = 4$ rollouts per input and clip parameter
 437 $\epsilon_{\text{clip}} = 0.2$. No KL penalty is applied (as outlined in Section B). We use the Adam [18] optimizer
 438 with learning rate $\eta_{\text{pre}}=10^{-4}$. Test statistics are estimated using 640 new samples.

439 C.2 Number Multiplication

440 **Architecture** We use the GPT2 [30] architecture which contains 124,439,808 parameters and the
 441 default GPT2 tokenizer that includes a vocabulary of size 50,257 and utilizes Byte Pair Encoding [32].

442 **Pre-training** We consider datasets with $p_{\text{cot}} \in \{0.1, 0.25, 0.5, 1\}$. The dataset and the pre-training
 443 code is based on the public repository of [8]. We train with the next-token prediction objective applied
 444 to the chain of thought (if it exists) and the answer tokens of the sequence. We use the AdamW [21]

optimizer with learning rate $\eta_{\text{pre}}=5 \cdot 10^{-5}$, batch size $B_{\text{pre}}=32$, and maximum gradient norm equal to 1.0. We train for 15 epochs for the 4×4 task and 50 epochs for 5×5 and 7×7 . All 3 training datasets consist of 808,000 unique samples. We train each model (3 random seeds per task) in a single GPU (NVIDIA A100-SXM4-80GB). For 4×4 , each run takes approximately 3-5 hours to complete (depending on the value of p_{cot}). For 5×5 and 7×7 , the completion time ranges in 21-38 hours.

Post-training We train with GRPO with an end-to-end reward function (that is, 1 if the tokens generated after ##### correspond to the correct answer³ and 0 otherwise). We use the AdamW [21] optimizer with learning rate $\eta_{\text{post}}=3 \cdot 10^{-6}$ (and Huggingface’s Transformers’ [47] default rest of hyperparameters), batch size $B_{\text{post}}=16$, group size (for GRPO) equal to 4, 2 steps of gradient accumulation (which makes the effective batch size equal to 32), no KL penalty, $\epsilon_{\text{clip}} = 0.2$. We use generation temperature $\tau_{\text{RL}} = 1.0$. We use the TRL library for efficient post-training [43]. We found it necessary to implement a wrapper around the model forward’s function to force the model to be in evaluation mode during the generation of the rollouts for RL (that is, to turn off any sources of non-determinism in the model, such as dropout layers). Otherwise, under TRL’s default implementation, the model was not able to generate correct responses, even during the very first RL iteration. We perform post-training for each random seed in a single GPU (NVIDIA A100-SXM4-80GB) which concludes in about 3.5 hours.

Test accuracy is estimated using a set of 1000 unseen pairs of numbers.

Estimate of total GPU hours needed to reproduce the pre-training and post-training results: 900

C.3 GSM8K

Our code is based on the torchtune library [41].

Architecture We use the Llama 3.2 3B base model for our GSM8K experiments.

We split each dataset into equal parts, perform supervised fine-tuning (SFT) on the first half with the next-token prediction objective applied to the answer portion of the sequence, and then switch to RL on the second half.

Supervised fine-tuning We train with the next-token prediction objective applied to the chain of thought and answer tokens of the sequence. We structure the data with preamble prompt (not calculating loss): "A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process here</think> <answer>answer here</answer>. User: {question} Assistant: " and structure the following sequence as: <think>cot</think> <answer>answer</answer>, where cot is everything that exists before characters ##### in the original GSM8k dataset. As before, we drop the cot from each sample with probability p_{cot} .

We use the AdamW [21] optimizer with learning rate $\eta_{\text{sft}}=10^{-5}$, batch size $B_{\text{sft}}=256$, and bf16 precision. We train for 30 epochs. We consider datasets with $p_{\text{cot}} \in \{0.1, 0.25, 0.5, 1\}$. We use PyTorch for distributed training. Each run is performed in 8 GPUs (NVIDIA A100-SXM4-80GB) and takes 1.5 hour to complete.

Post-training We train with GRPO with a reward function that has stepwise structure: 5 points for the presence of the answer tag, 5 points for the presence of the thinking tag, 20 points if it contains the correct answer in part of the response, and 100 points for a correct answer at the appropriate place. We found it important using the math-verify module of Huggingface for simplifying mathematical expressions and accurately rewarding generations. We use the AdamW [21] optimizer with learning rate $\eta_{\text{post}}=10^{-5}$, cosine learning rate scheduler, batch size $B_{\text{post}}=1$, group size (for GRPO) equal to 32, no KL penalty, $\epsilon_{\text{clip}} = 0.2$. We use generation temperature $\tau_{\text{RL}} = 1.0$. We perform post-training

³That is, “complete number match”.

in 8 GPUs (NVIDIA A100-SXM4-80GB) and each run concludes in 31-35hours and 20hours for $p_{\text{cot}} = 0$.

Test statistics are estimated on the test set of GSM8k using 4 generations per question and taking the average.

Estimate of total GPU hours needed to reproduce the pre-training and post-training results: 11k

C.4 MATH

Our code is based on the torchtune library [41].

Architecture We use the Llama 3.2 3B, and 3.1 8B instruct models.

We split each dataset into equal parts⁴, perform supervised fine-tuning (SFT) on the first half with the next-token prediction objective applied to the answer portion of the sequence, and then switch to RL on the second half.

Supervised fine-tuning We train with the next-token prediction objective applied to the chain of thought and answer tokens of the sequence. We structure the data with preamble prompt (not calculating loss):

```
<|begin of text|><|start header id|>system<|end header id|>
Cutting Knowledge Date: "
"December 2023
Today Date: 26 Jul 2024
A conversation between User and Assistant. The user "
"asks a question, and the Assistant solves it. "
"The assistant first thinks about the reasoning "
"process in the mind and then provides the user with the answer. "
"The reasoning process and "
"answer are enclosed within <think></think> "
"and <answer></answer> tags, respectively, i.e., "
"<think>reasoning process here</think> <answer>answer "
"here</answer>. Inside the answer tag, put only "
"the answer and no additional commentary. <|eot id|><|start "
"header id |>user<|end header id|> {question} "
"<|eot id|><|start header id|>assistant<|end header id|>
```

and structure the sequence as: $\langle \text{think} \rangle \text{cot} \langle \text{think} \rangle \langle \text{answer} \rangle \text{answer} \langle \text{answer} \rangle$, where cot is everything that exists in the original MATH sequence before the boxed answer. We drop the cot with probability p_{cot} .

We use the AdamW [21] optimizer with learning rate $\eta_{\text{sft}}=10^{-5}$, batch size $B_{\text{sft}}=24$ for the 3.2 3B model and $B_{\text{sft}}=16$ for the 3.1 8B model, and bf16 precision. We train for 20 epochs. We consider datasets with $p_{\text{cot}} \in \{0.1, 0.25, 0.5, 1\}$. We use PyTorch for distributed training. Each run is performed in 8 GPUs (NVIDIA A100-SXM4-80GB) and takes 1.5 hour to complete.

Post-training We train with GRPO with a reward function that has stepwise structure: 5 points for the presence of the answer tag, 5 points for the presence of the thinking tag, 20 points if it contains the correct answer in part of the response, and 100 points for a correct answer at the appropriate place. We found it important using the math-verify module of Huggingface for simplifying mathematical expressions and accurately rewarding generations. We use the AdamW [21] optimizer with learning rate $\eta_{\text{post}}=10^{-5}$ (and Huggingface’s default rest of hyperparameters), batch size $B_{\text{post}}=1$, group size (for GRPO) equal to 20, 2 steps of gradient accumulation (which makes the effective batch size equal to 2), no KL penalty, $\epsilon_{\text{clip}} = 0.2$. We use generation temperature $\tau_{\text{RL}} = 1.0$. We perform post-training in 8 GPUs (NVIDIA A100-SXM4-80GB) for 300 steps which concludes in about 14-29 hours.

⁴We split the datasets, since they are of small size and we perform multi-epoch training during SFT.

D Additional experimental results

D.1 More experiments on main parity setting

Explanation of learning mechanisms in main setting *On pre-training vs pre-training and post-training:* Observe that if post-training starts too early, then it might not lead to a generalizing model despite the length increase. However, GRPO, when started from later checkpoint, improves performance rapidly, and the number of post-training samples required for generalization is a miniscule fraction of the overall number of samples. On the other hand, mere next-token prediction does not result in a generalizing model (and the median greedy response remains short) even with access to $\sim 10^7$ training samples. This demonstrates the enormous sample-complexity gap between the two approaches for this task. Note that when the input dimension d is smaller, pre-training eventually does lead to a well-generalizing model (if we optimize for sufficiently long) and post-training does not seem strictly necessary (see Figure 10 for $d \in \{11, 12\}$). However, for larger values of d , the amount of SGD iterations required is unreasonably large. In other words, the transformers’ inability to succeed during pre-training **does not stem from insufficient model capacity** (small number of parameters or lack of depth⁵), but it is rather because next-token prediction on this distribution requires many **samples** to lead to a good predictor. In fact, as the complexity of the task increases, so does the advantage of reinforcement learning after next-token prediction over mere next-token prediction (Figure 11). Appendix D.1 contains more experimental results for various model and task hyperparameters.

On different speeds of learning & growth of test-time compute: We now describe what makes reinforcement learning so effective and why post-training leads to length increase in our simple setting. First, notice the curves in Figure 2 that correspond to sampling with temperature 1 during pre-training. Even when $p_{\text{cot}} < 1/3$, the accuracy of the models at the end of pre-training is greater than 50% – Figure 2 (Left). In fact, it is equal to $\frac{p_{\text{cot}}+1}{2} \times 100\%$ for all cases of p_{cot} . Furthermore, notice in the length plot of the same figure (Figure 2 (Center)) that the models are *calibrated* with respect to length, i.e., they generate long responses with probability p_{cot} . These two simple observations directly suggest that the models learn the two parts of the mixture “in parallel” during pre-training: after a few training iterations, when prompted with d bits x_1, \dots, x_d , the models learn to generate long and correct responses containing the parity $\prod_{i=1}^d x_i$ with probability p_{cot} and short responses with probability $1 - p_{\text{cot}}$. As mentioned before, learning from the short sequences requires learning the parity of d bits in a single prediction step, which belongs to a class of computationally difficult problems [17] and is believed to be hard for standard neural networks to learn in practice [34, 37]. As a result, we do not expect the model to learn using any reasonable amount of samples, and hence it resorts to short, random guessing with probability $1 - p_{\text{cot}}$. Learning from long demonstrations, on the other hand, can be performed efficiently [22]. This asymmetric learning difficulty leads to accuracy equal to $p_{\text{cot}} \times 100\% + (1 - p_{\text{cot}}) \times 50\% = \frac{p_{\text{cot}}+1}{2} \times 100\%$.

The learning process during pre-training sets the stage for what follows in post-training. It is perhaps simpler to understand the RL dynamics in the case of the STaR algorithm with the chain-of-thought correctness reward r_{cot} (cyan lines in Figure 3). The objective of (6) implements next-token prediction solely on model-generated sequences which contain a correct chain of thought and a correct final answer. If, at the end of pre-training, the probability of a long, correct generation is p_{cot} and the probability of a short, correct one is $(1 - p_{\text{cot}})/2$ (as suggested by Figure 2), then the model fits a data mixture which contains both long and short, correct sequences with proportions equal to $p_1 := \frac{p_{\text{cot}}}{p_{\text{cot}} + (1-p_{\text{cot}})/2}$ and $\frac{(1-p_{\text{cot}})/2}{p_{\text{cot}} + (1-p_{\text{cot}})/2}$, respectively. If, further, the model succeeds in fitting them, then the conditional distribution of model generations at the end of the first round of STaR has the same weights, provided the model remains length-calibrated and did not manage to learn from the short sequences. Continuing inductively, at the start of the n ’th round, we effectively sample from distribution $\mathcal{D}(p_n)$, where $p_0 = p_{\text{cot}}$ and $p_n = \frac{2p_{n-1}}{1+p_{n-1}}$ converges to 1 exponentially fast. Once the effective coefficient p_n becomes larger than $\frac{1}{3}$ (which happens when accuracy with temp. 1 exceeds $\frac{1/3+1}{2} = \frac{2}{3}$ – red dashed line in Figure 3), the model starts generating long, correct responses with greedy decoding, thus generalizing perfectly at the task. This is the mechanism that causes length increase in model response, which in turns allows the model to learn to generalize.

⁵A shallow transformer can approximate the parity function for any input dimension d ; uniform attention for aggregating the bit values and a 2-layer MLP for computing their XOR – see Lemma 6 in [20].

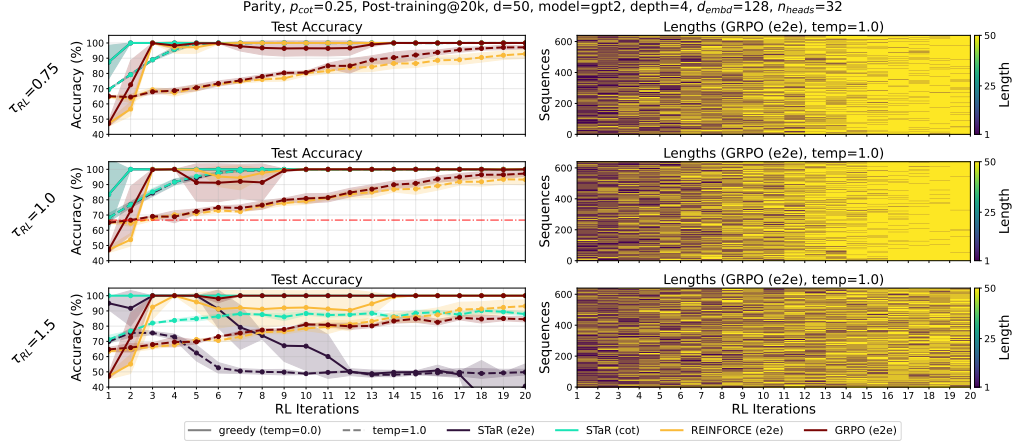


Figure 3: **Post-training of transformers on mixture of long and short sequences encoding the parity of d bits with various RL methods and generation temperatures (τ_{RL}).** *Left:* Test accuracy during the course of post-training with greedy decoding (solid lines) and sampling with temperature 1 (dashed lines) for various RL methods. Figure shows average and 1 standard deviation across 3 seeds. *Right:* Length of generated response (sampled with temperature 1) during the course of a post-training run (GRPO with end-to-end reward) for 640 test inputs, after 20k pre-training iterations. **Note:** The sample size n of each RL iteration differs amongst the RL algorithms: $n=64$ for GRPO, REINFORCE and $n=3,200$ sequences for STaR.

In Appendix D.2, we consider a variation of our setting, where the data include a third type of sequence consisting of a partial cot. We demonstrate how this affects length distributions at the end of RL, and the effectiveness of length penalties during RL as a function of prior pre-train iterations.

Model hyperparameters ablation We present aggregated results for pre-training and post-training on $\mathcal{D}(p_{\text{cot}})$, $p_{\text{cot}}=0.25$ for GPT2 and Mistral architectures of varying embedding dimension and number of heads in Figure 4 (depth $L=2$), Figure 5 (depth $L=4$) and Figure 6 (depth $L=8$). Post-training uses GRPO with end-to-end reward function r_{e2e} and sampling temperature $\tau_{\text{RL}}=1.0$ for all configurations. Notice that the Mistral architecture exhibits more unstable learning (pre-training) than GPT2. We observe that pre-training alone does not induce a model capable of consistent generalization in any of the cases. We plot number of training sequences on the x-axis (as opposed to, say, SGD iterations to allow a simpler comparison with post-training). For the number of post-train samples, we take the multiplicity of GRPO samples into consideration.

Comparison of RL algorithms for more values of p_{cot} In Figure 3, we presented test accuracy and response length during post-training for all RL algorithms (STaR, REINFORCE, GRPO) and for a few different values of sampling temperature used for generating training sequences for $p_{\text{cot}}=0.25$. We show now additional results for distributions where data are even more rare ($p_{\text{cot}}=0.1$ and $p_{\text{cot}}=0.05$). In Figure 7, we observe that all RL algorithms induce generalization for most values of sampling temperature (post-training starts after 20k pre-training iterations) when $p_{\text{cot}}=0.1$. On the other hand, when p_{cot} is even smaller ($p_{\text{cot}}=0.05$), post-training after 20k iterations does not reliably lead to well-generalizing models for any of the RL algorithms (Figure 8). If, instead, we consider more pre-training iterations (50k) before the start of post-training, then we observe greater probability of success with all RL algorithms for reasonable (≈ 1) sampling temperatures (Figure 9). This illustrates, amongst others, that the learning behaviors of STaR, REINFORCE and GRPO are similar in our simple setting.

Smaller input dimension In Figure 10, we demonstrate that pre-training sometimes leads to a generalizing model if we wait long enough (example seeds shown for $d=11$ and $d=12$). However, as d increases, this waiting time becomes prohibitive.

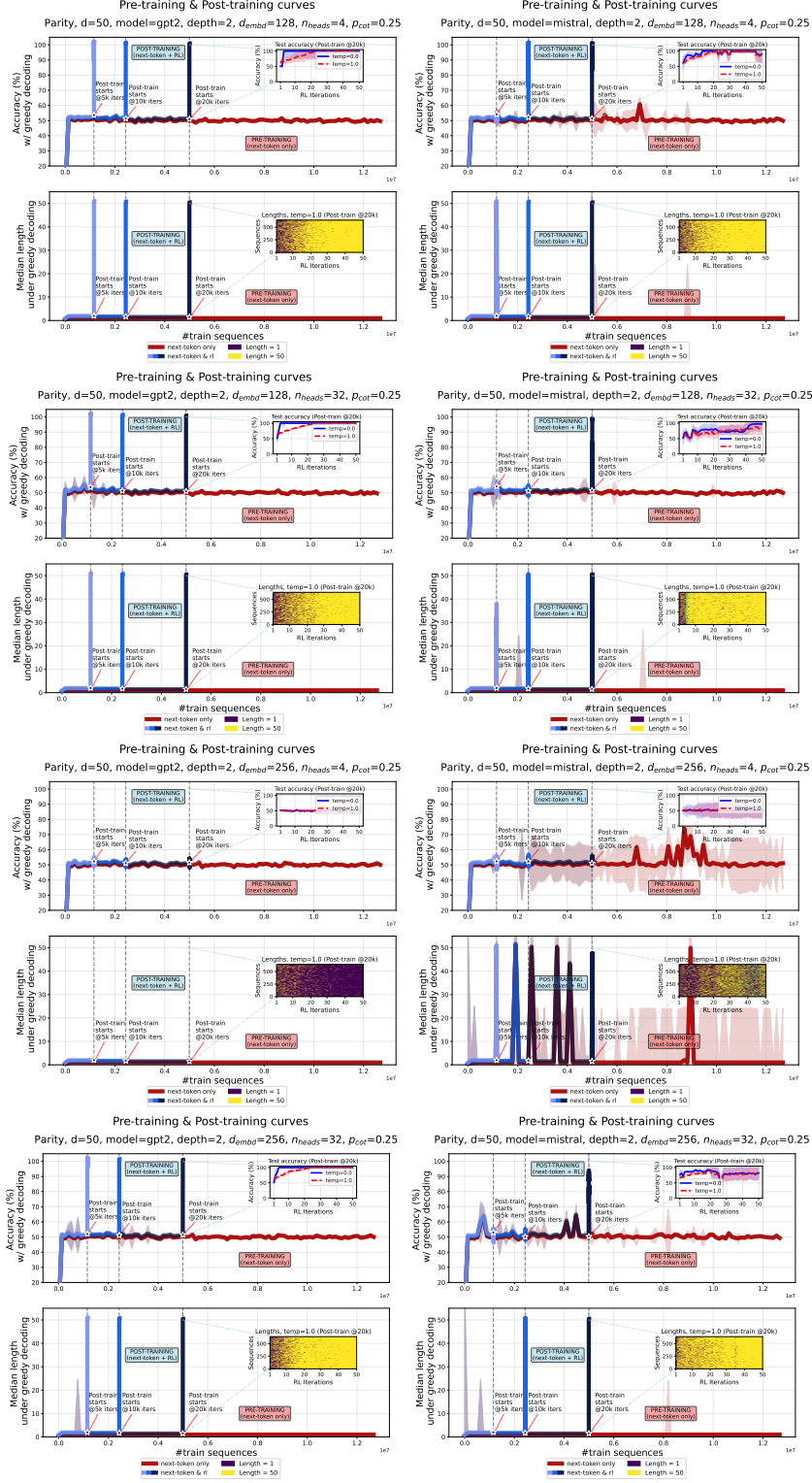


Figure 4: Pre-training and post-training curves combined on the parity task for GPT2 and Mistral architectures. We vary the embedding dimension and the number of heads. Depth $L=2$.

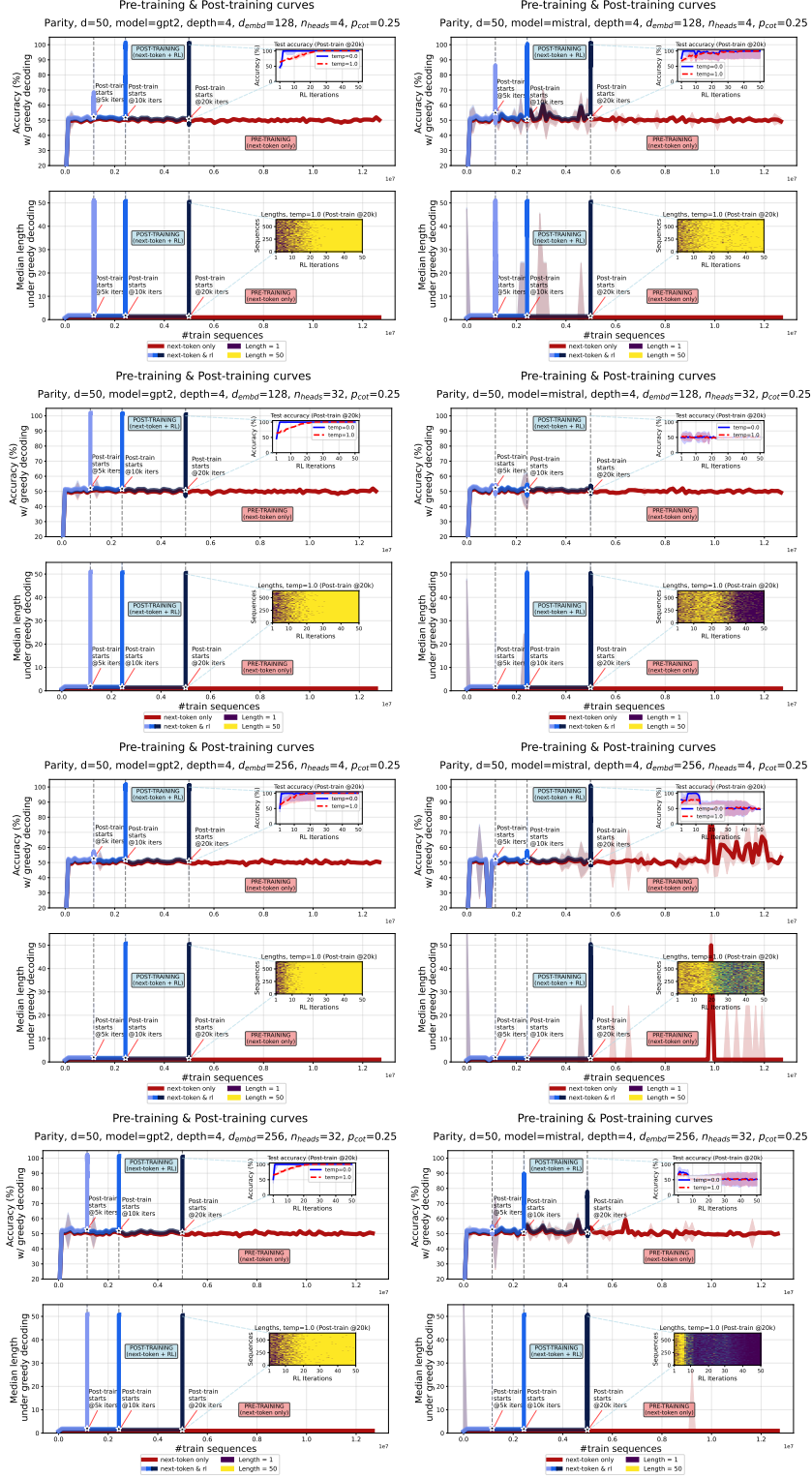


Figure 5: Pre-training and post-training curves combined on the parity task for GPT2 and Mistral architectures. We vary the embedding dimension and the number of heads. Depth $L=4$.

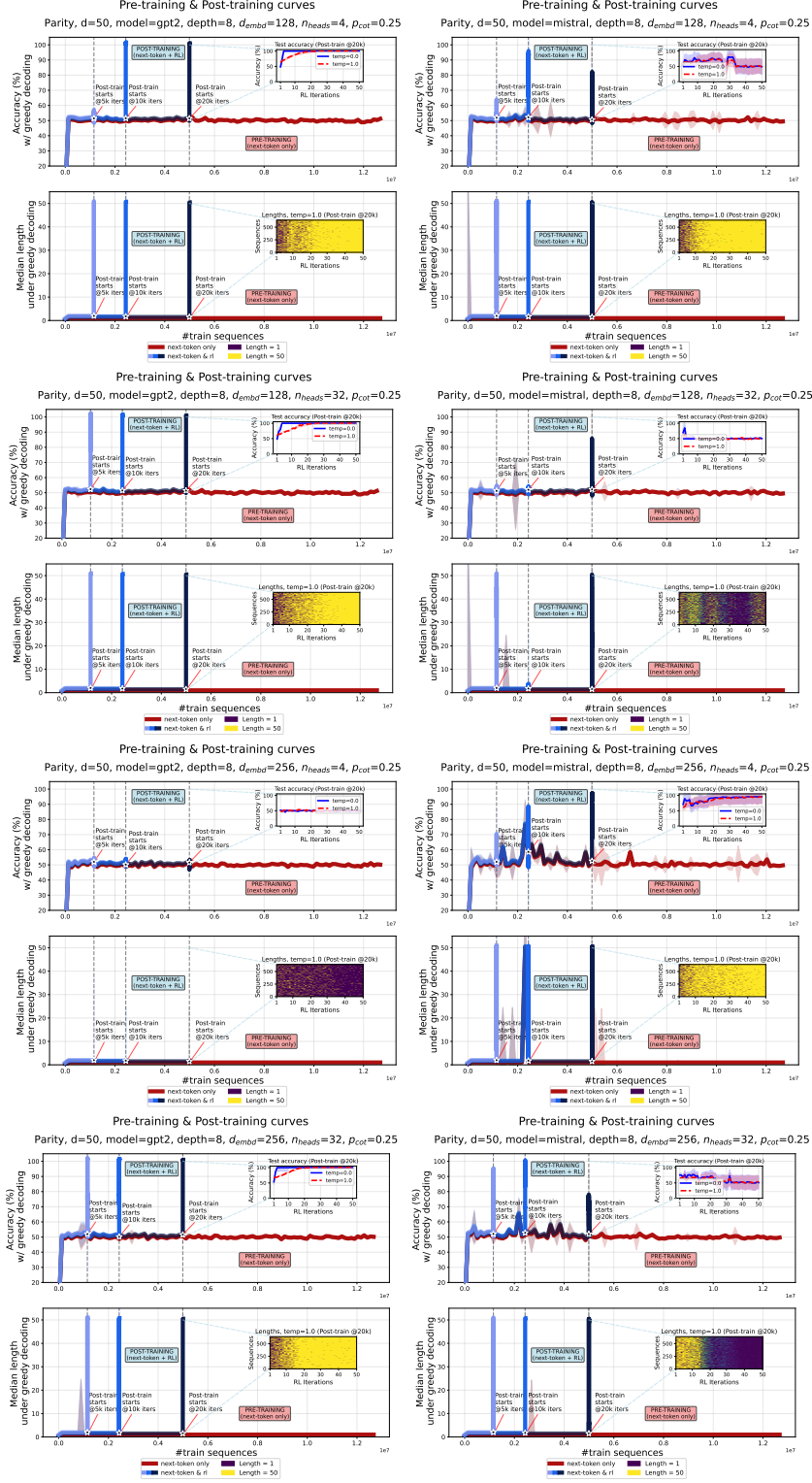


Figure 6: Pre-training and post-training curves combined on the parity task for GPT2 and Mistral architectures. We vary the embedding dimension and the number of heads. Depth $L=8$.

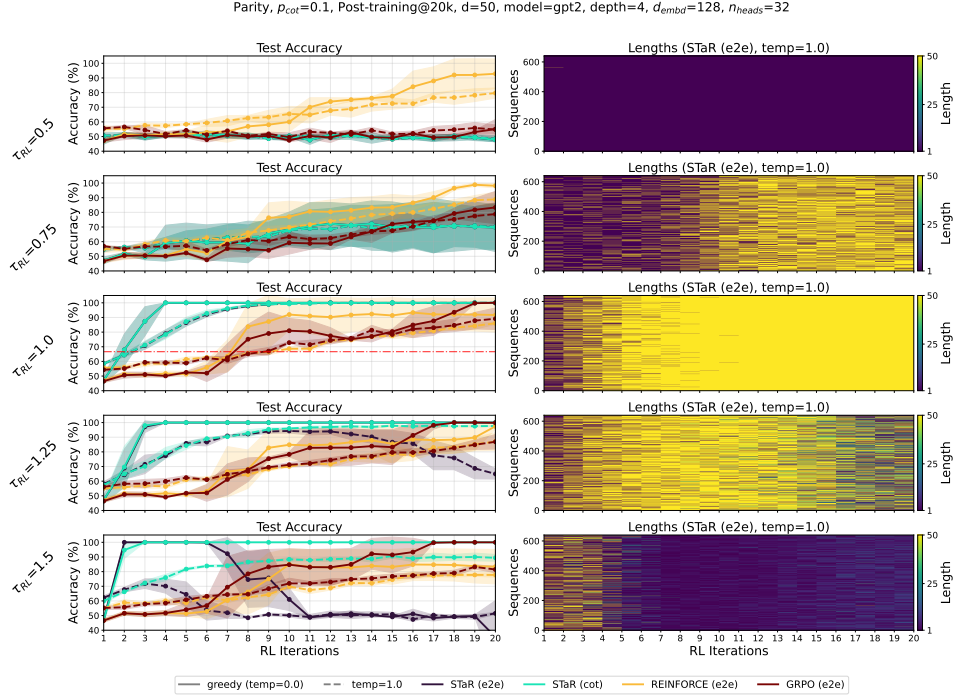


Figure 7: **Post-training of transformers on mixture of long and short sequences encoding the parity of d bits with various RL methods and generation temperatures (τ_{RL}).** Mixture coefficient: $p_{cot}=0.1$. Pre-training iterations: 20k.

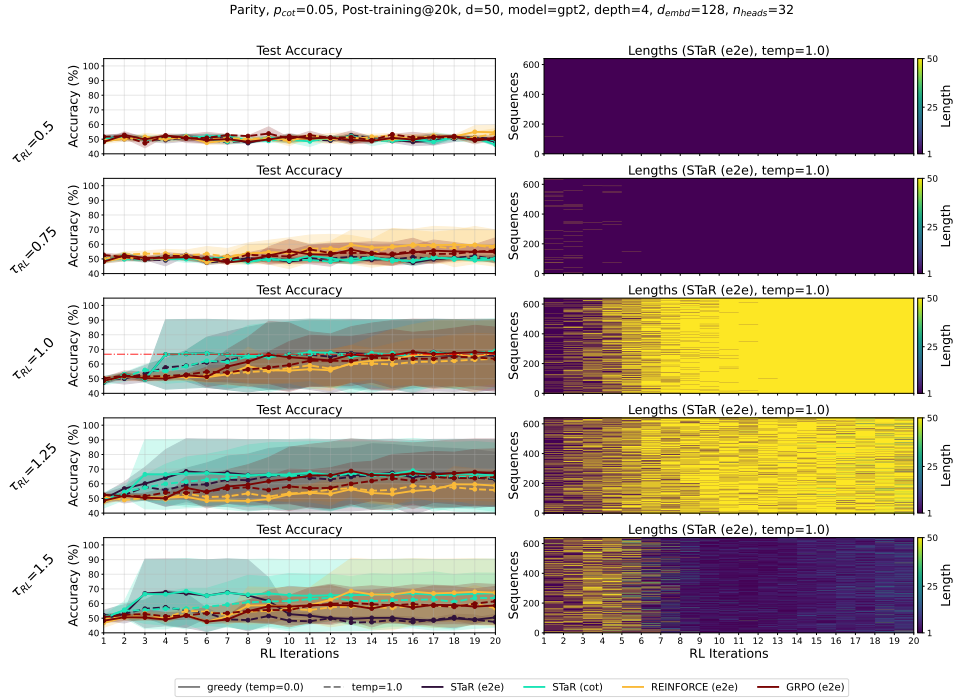


Figure 8: **Post-training of transformers on mixture of long and short sequences encoding the parity of d bits with various RL methods and generation temperatures (τ_{RL}).** Mixture coefficient: $p_{cot}=0.05$. Pre-training iterations: 20k.

Parity, $p_{cot}=0.05$, Post-training@50k, $d=50$, model=gpt2, depth=4, $d_{embd}=128$, $n_{heads}=32$

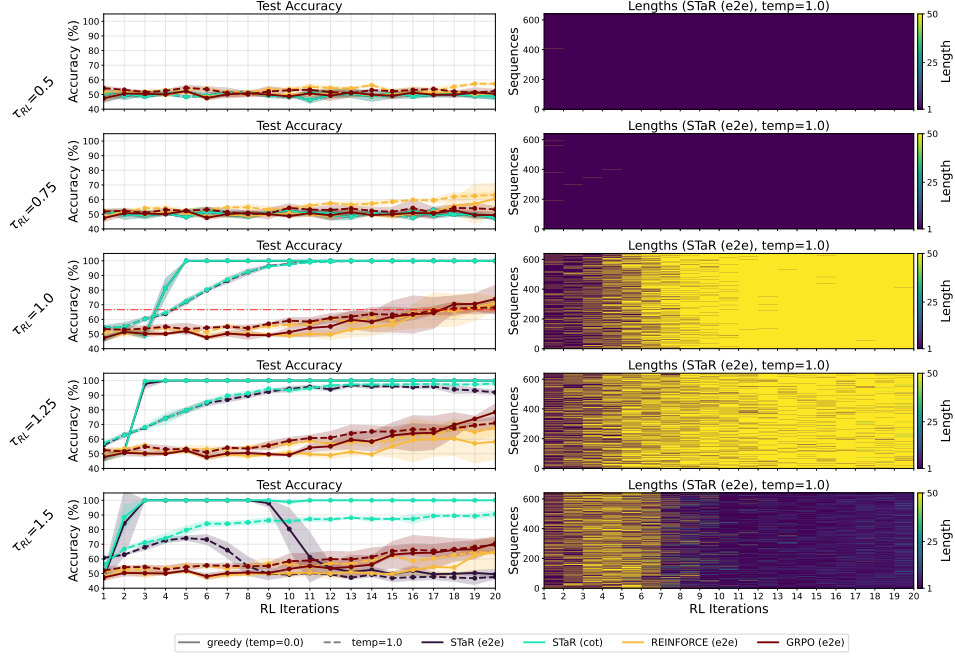


Figure 9: **Post-training of transformers on mixture of long and short sequences encoding the parity of d bits with various RL methods and generation temperatures (τ_{RL}).** Mixture coefficient: $p_{cot}=0.05$. Pre-training iterations: 50k.

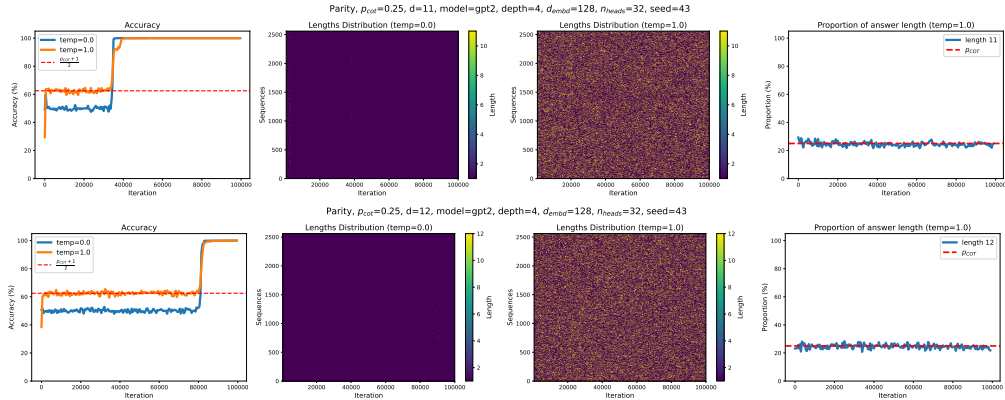


Figure 10: **Pre-training on the parity task leads to a generalizing model after many iterations.** The average response is short (length equal to 1). *Top row: $d=11$. Bottom row: $d=12$.*

Sample complexity gap We design an experiment to try estimating the sample complexity gap between the two paradigms (next-token only vs next-token followed by RL). We consider a GPT2 architecture of depth 4, number of heads 32 and embedding dimension equal to 128. We fix $p_{\text{cot}}=0.25$ and we vary the input dimension $d \in \{6, 8, 10, 11, 12\}$. We train for at most 100,000 iterations (of batch size 256) and we consider switching to post-training (GRPO with e2e reward) at several checkpoints. We consider a checkpoint to be successful if its test accuracy is greater than 95%. In Figure 11, we show the gap between the number of samples required for any pre-trained checkpoint to generalize vs the number of samples required by any pre- and post-trained model (combined) for several input dimensions d . For the number of post-train samples, we take the multiplicity of GRPO samples into consideration. This gap seems to be growing exponentially fast. Some limitations of this simple experiment: the pre-trained checkpoints from which we consider starting post-training are at iterations $\{1000, 1500, 2000, 2500, 3000, 5000, 10000, 15000, 20000, 30000, 50000, 100000\}$. Also, for $d=12$, only one of three seeds ended up generalizing during pre-training (so by definition we could only calculate the gap for that seed and we discarded the rest).

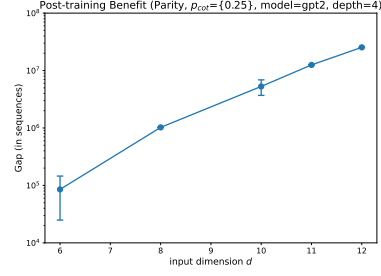


Figure 11: Sample complexity advantage of pre-training & post-training over mere pre-training vs input dimension. Note the logarithmic scale in the y-axis. The figure shows mean and one standard deviation across 3 random seeds.

D.2 Partial chain of thought

The purpose of this subsection is to relax our main setting, and consider a more complex variation which might better capture “real-world” data. In particular, we consider a setting, where the data include a third type of sequence consisting of a medium-sized chain of thought, in addition to short and long sequences.

We consider the distribution $\mathcal{D}_3(p_{\text{cot}}, p_{\text{odd}})$ over $\{\pm 1\}^d \times \{\pm 1, \langle \text{EOS} \rangle\}^*$, parameterized by $p_{\text{cot}}, p_{\text{odd}} \in (0, 1)$, such that: $x_1, \dots, x_d \sim \text{Rad}(1/2)$ and

$$(y_1, \dots, y_{d+1}) = \begin{cases} \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right), & \text{w.p. } p_{\text{cot}}, \\ \left(x_1, x_1 x_2 x_3, x_1 x_2 x_3 x_4 x_5, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right), & \text{w.p. } p_{\text{odd}}, \\ \left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right), & \text{w.p. } 1 - p_{\text{cot}} - p_{\text{odd}}. \end{cases} \quad (11)$$

The medium-sized sequence consists of the same random variables as before in input positions 1 to d , while the chain of thought skips the intermediate computations that involve partial parities ending at an even position (with the exception of the final parity, if d is even). This results in sequences that contain a *partial* chain of thought with omitted reasoning steps. The purpose of this sequence is to provide the model with some samples of intermediate complexity between the two extremes of $\mathcal{D}(p_{\text{cot}})$. Indeed, learning to predict each term of the chain of thought after the first position requires a *leap* [1] of order 2, as opposed to either leap 1 or d for short and long sequences, respectively. For instance, predicting $x_1 x_2 x_3$ from the history of the sequence (x_1, \dots, x_d, x_1) is a leap of order 2. The leap between terms characterizes the additional sample complexity for learning each (sub)function.

We repeat pre-training and post-training on this distribution and present results in Figure 12 for $d=15$ and $(p_{\text{cot}}, p_{\text{odd}})=(0.1, 0.1)$.

Pre-training staircase As before, improvement during pre-training happens in stages: early at training, model accuracy under sampling with temperature 1 is around 50%, yet it suddenly jumps to $\frac{p_{\text{cot}}+1}{2} \times 100\%$ as the model learns from the long demonstrations. Later in training, there is another phase transition indicating the model has learned the medium-sized sequences, as the accuracy leaps to $\frac{p_{\text{cot}}+p_{\text{odd}}+1}{2} \times 100\%$. As before, if $p_{\text{cot}}, p_{\text{odd}}$ are not large enough, then the models under greedy decoding do not generalize in reasonable time.

Pre-training checkpoints & length penalties In this setting, the point when we switch to post-training, can not only affect the eventual success of RL, but also the length distribution of the model responses at the end of it:

Parity, Medium chain of thought, $(p_{cot}, p_{odd})=(0.1, 0.1)$, $d=15$, $depth=4$, $seed=44$

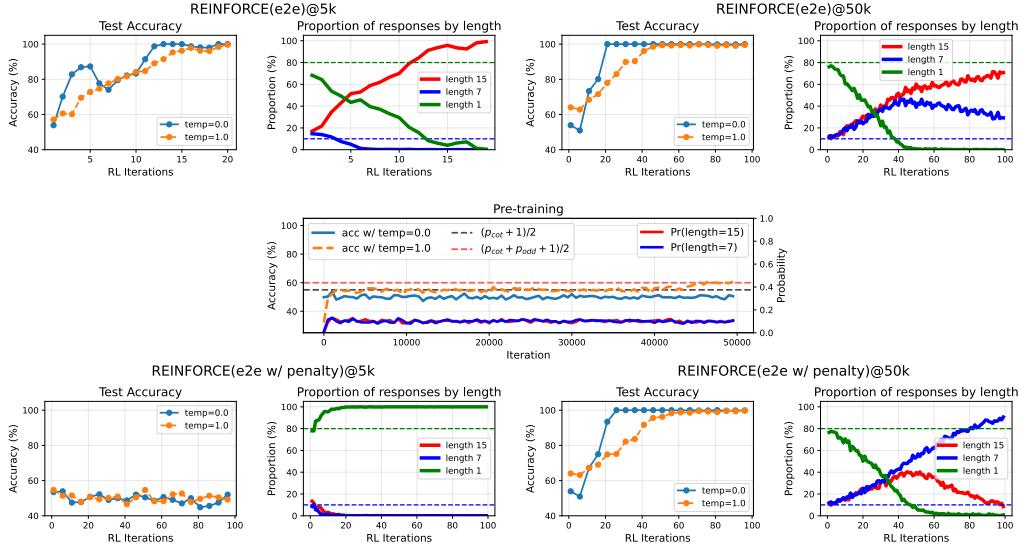


Figure 12: **Parity with partial chain of thought data and effectiveness of length penalties.** Post-training experiments using REINFORCE with (bottom row) and without (top row) a length penalty on pre-trained models that correspond to the training shown in the middle. Notice that leaps in accuracy during pre-training control length distributions at the end of post-training and success of length-penalized RL.

- Post-training at 5k iterations yields a model that generalizes, yet it learns to almost exclusively generate long responses (Figure 12, top left).
- Post-training at 50k iterations, on the other hand, results in a model that not only manages to reach 100% test accuracy but also has more diversity in its length distribution, generating both long and medium-sized responses (Figure 12, top right).

The previous observation is both good and bad news: good news, as the 50k checkpoint ends up being cheaper at inference than the post-trained 5k counterpart; but bad news, as the model could have been even faster at inference without sacrificing accuracy. Indeed, after 50k pre-training iterations the model has learned to generate correct medium-sized responses, hence, in principle, it need not learn from the long self-generated ones. To test this, we consider reinforcement learning with a length-penalized reward. In particular, we consider the following length-penalized reward:

$$r_{e2e, \lambda_{len}}(\mathbf{x}, y) = \mathbb{1} \left\{ y[-1] = \prod_{i=1}^d x_i \right\} - \lambda_{len} \frac{|y|}{d}, \lambda_{len} \in [0, 1]. \quad (12)$$

We use $\lambda_{len}=0.4$. As we confirm in Figure 12 (bottom row), a length penalty applied during post-training of a late checkpoint is effective, as it leads to a model that is both accurate and fast. On the contrary, if we apply the same length penalty at an earlier checkpoint, post-training fails. This is because the pre-trained model has not managed to learn from the medium-sized sequences early in pre-training.

E Experiments with Mathematical Reasoning Benchmarks

A shortcoming of the previous sections is that they are devoted to the study of a computationally shallow problem. The parity of d bits can be represented by a small, shallow transformer [20], yet the current post-training paradigm targets tasks that appear to require greater computational depth, involving sequences of dependent operations that cannot be collapsed into a single “step”. To accommodate such tasks, the iterative application of the transformer in the form of the autoregressive

generation process enhances the computational depth of the model. Without this generated chain-of-thought, models may fail to approximate solutions to certain tasks as context length grows [10, 25, 19]. Can reinforcement learning after next-token prediction be effective when learning from mixture distributions that encode such deeper problems? In other words, can RL be as effective when representation, in addition to learning, is also an issue?

To study this question, we now consider less idealized settings (including pre-trained models) and computationally deeper tasks, such as the multiplication of 5-digit numbers and mathematical reasoning benchmarks.

E.1 Number Multiplication

First, we study the task of multiplying n -digit numbers. It has been reported that even LLMs as capable as GPT4 [2] have struggled with accurate multiplication of 4-digit numbers [36] without the use of an external calculator program⁶. We encode the task in sequences of characters, including the n digits of the multiplier and multiplicand (both in reversed order), the execution trace of the grade-school multiplication algorithm, and the final answer. Prior work has reported that reversing the input order helps transformers with absolute positional embeddings generalize better on arithmetic tasks [36]. Following our previous setup, we construct datasets that contain the n digits of the multiplier and multiplicand, together with the execution trace of the grade-school multiplication algorithm and the final answer, with probability p_{cot} , and just question, answer with probability $1 - p_{\text{cot}}$ (Figure 13).

We pre-train randomly initialized GPT2 transformers on either 4×4 , 5×5 or 7×7 datasets for multiple epochs over a training set of 808k examples using the next-token prediction objective. At various checkpoints, we switch to reinforcement learning (with GRPO) and compare performance during pre-training vs post-training. Full experimental details are provided in Appendix C.2.

Figures 15 to 17 show pre-training results for all values of $n \in \{4, 5, 7\}$ and $p_{\text{cot}} \in \{0, 0.1, 0.25, 0.5, 1.0\}$. When $p_{\text{cot}}=1.0$ (i.e., the algorithm’s trace is included in all samples), models during pre-training quickly learn to answer correctly with long responses, leveraging the intermediate calculations present in the training data. By contrast, when p_{cot} is small, models struggle to generalize: performance under greedy decoding remains close to 0% throughout training. In some runs, we observe sudden increase in accuracy under sampling with temperature 1.0, jumping to approximately $p_{\text{cot}} \times 100\%$. This outcome is consistent with the parity experiments: the objective enforces length calibration and, once the model learns from long sequences, its output becomes a mixture of short random guesses (with probability of success $10^{-2 \times n}$) and long correct responses (with probability of success 1). Similar to the parity experiments, we observed unstable behavior under greedy decoding when training with p_{cot} close to the critical threshold (which is close to 0.5 here).

After switching to reinforcement learning (Figure 14 (left) for $(n=5, p_{\text{cot}}=0.25)$ and Figure 18 for the rest of the values), the model’s accuracy improves rapidly and the average response length increases, provided the pre-training checkpoint has developed some correlation with the target. These observations hold for most values of n and p_{cot} . For the most challenging setting we considered (largest n , smallest p_{cot} – $(n, p_{\text{cot}}) = (7, 0.1)$), the model failed to learn from long sequences under any of the 3 random seeds, even after 38 GPU hours, preventing post-training from being successful in this case. Note that this section provides an example of a task where the capability obtained during RL (accuracy under greedy decoding on task of multiplication) appears to be almost absent during pre-training (accuracy goes from ≈ 0 to 100%) – for example, when p_{cot} is equal to 0.1 or 0.25.

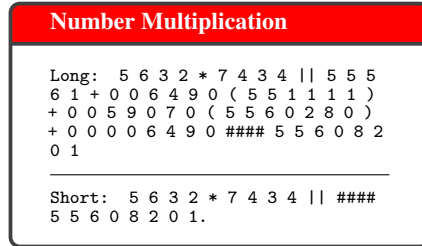


Figure 13: Example of a 4x4 sequence, encoding 2365×4374 (digits appear in reverse order). Top row: long format with full sequence. Bottom row: short format without chain of thought.

⁶Otherwise, the problem reduces to copying input (from the calculator program) to output and transformers are well-suited for it [14].

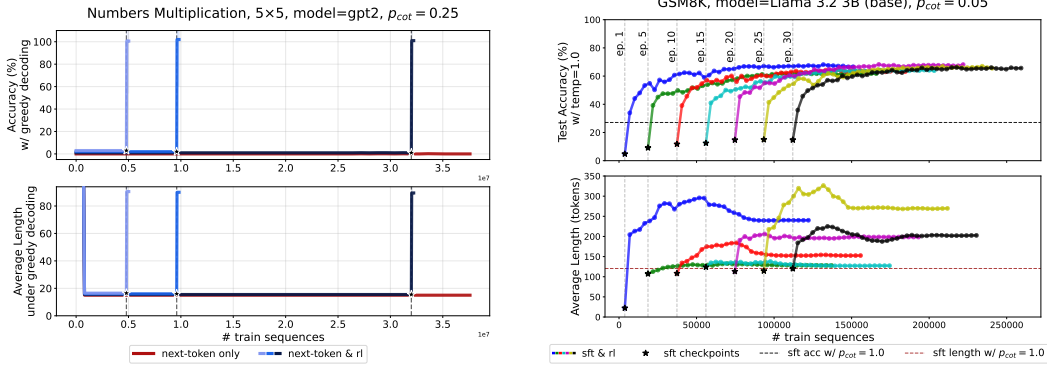


Figure 14: **Advantage of next-token prediction followed by reinforcement learning over mere next-token prediction** in (Left) multiplying two 5-digit numbers with a GPT2 model trained from scratch, and in (Right) solving grade-school math problems with Llama 3.2 3B (base). Left: Red line: pre-training on a 5x5 dataset with $p_{cot}=0.25$ using next-token prediction. Blue lines: the same pre-training runs, each followed by GRPO from a different checkpoint. Top: median test accuracy under greedy decoding (3 seeds). Bottom: median average response length (median over seeds, averaged over sequences). Right: \star markers: supervised fine-tuning checkpoints on GSM8k with $p_{cot}=0.05$ (epochs 1–30). Colored curves: GRPO post-training from these checkpoints. Training sequences are reused across epochs; during post-training, multiple generations per input are counted.

E.2 Grade School & High School Mathematics (GSM8K & MATH datasets)

Finally, we experiment with pre-trained LLMs and mathematical reasoning benchmarks. We train Llama pre-trained models [9] on variations of two standard mathematical benchmarks (GSM8k [6] and MATH [12]). We format the data by surrounding the chain of thought of each sample with special “thinking” tokens. As previously, the chain of thought is replaced by the empty string with probability p_{cot} , creating a mixture distribution of long and short reasoning sequences. We first perform supervised fine-tuning (SFT) on these mixture datasets, and then switch to RL with GRPO and a reward that assesses both correctness of the response and consistency with the specified data format. For experimental details, see Appendices C.3, C.4.

We plot results for Llama 3.2 3B (base) on GSM8k, $p_{cot}=0.05$, in Figure 14 (right). We show test accuracy and average response length for the SFT checkpoints (stars) and the post-training curves starting from these checkpoints. As a baseline, we also plot the best SFT model trained with full chain of thought data; the $p_{cot}=1.0$ runs can be found in Figure 19. Early in SFT, accuracy is low and the average length of the model response is much smaller than the baseline. As training with the next-token objective continues, accuracy improves and length increases, but accuracy does not reach the baseline and appears to saturate at a lower value. Once we switch to RL with GRPO, the model rapidly reaches baseline accuracy while increasing its average response length. Notice how few samples RL requires to reach the baseline, even when starting from the first epoch checkpoint. This sample count should be contrasted with the number of SFT updates needed to reach comparable accuracy. Furthermore, in Figures 22 to 24, we provide examples of model completions during RL for this first checkpoint. We observe that early on, the model respects the SFT format but primarily generates short responses (Figure 22). As RL progresses, the model learns to produce longer responses with greater probability (Figure 23), eventually reaching a point where it consistently generates correct elaborate responses that mimic the chain of thoughts in the long-form part of the training distribution (Figure 24). These training patterns are consistent with the mechanisms observed in the parity setting of the previous sections. In Figure 19, we repeat this experiment for various p_{cot} values.

Beyond the point where accuracy exceeds the SFT baseline, the model continues to improve and its response length grows even further. This phase of post-training differs from the situations observed in the previous sections and is likely due to the model leveraging pre-training data. Indeed, in Figure 25, we show that model generations at the end of RL are qualitatively different from the train sequences.

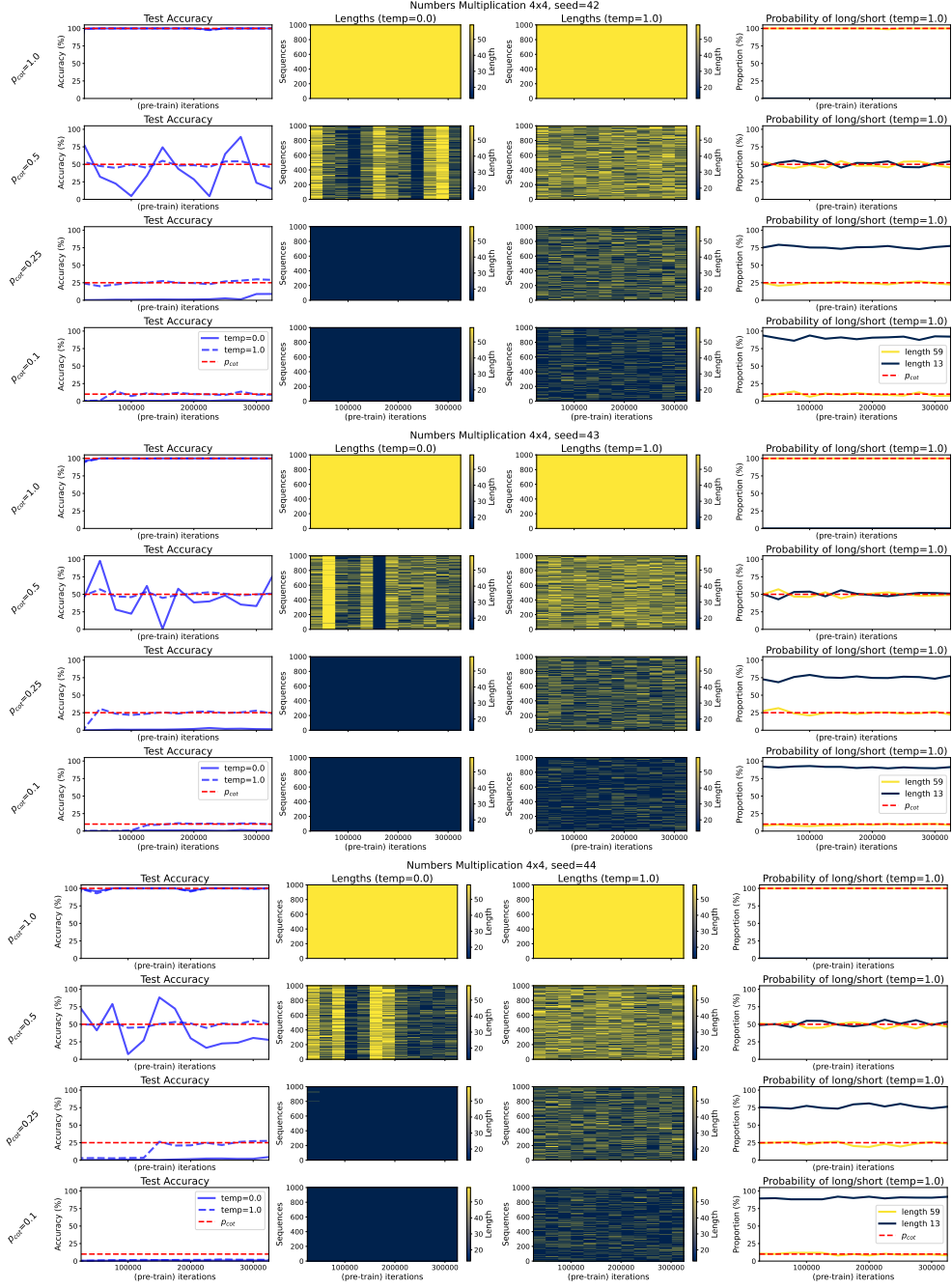


Figure 15: **Pre-training of GPT2 transformers on mixture of long and short sequences encoding the multiplication of 4-digits numbers for 3 different seeds and for various values of p_{cot} .** *Left:* Test accuracy during the course of pre-training, under greedy decoding and sampling with temperature 1. *Center, Left:* The length of the greedy response for 1000 test samples during the course of pre-training. *Center, Right:* The length of the sampled response (temperature of 1) for 1000 test samples during the course of pre-training. **Right:** The probability that the length of a model’s autoregressive generation equals the maximum or minimum length present in the training distribution.

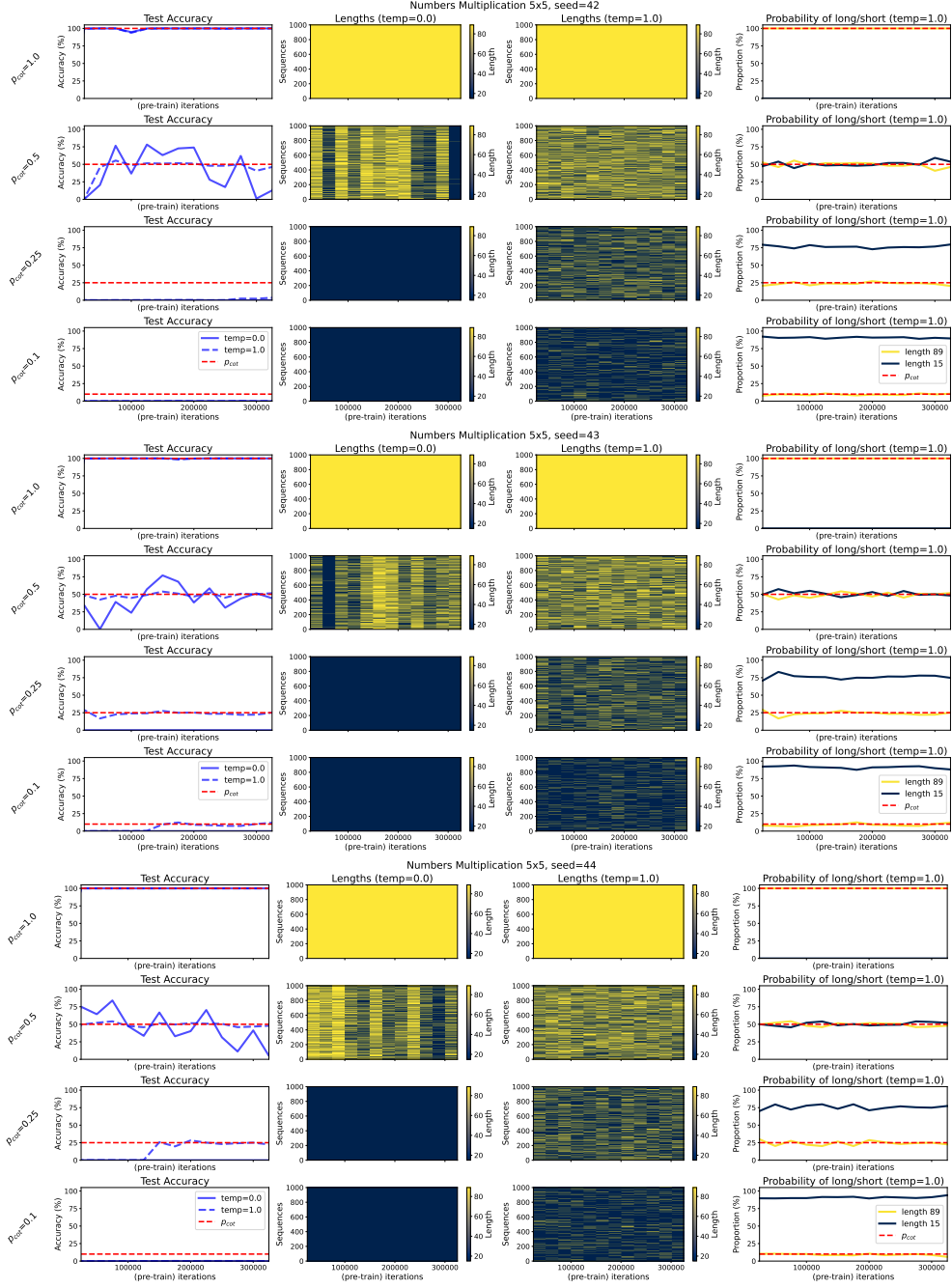


Figure 16: **Pre-training of GPT2 transformers on mixture of long and short sequences encoding the multiplication of 5-digits numbers for 3 different seeds and for various values of p_{cot} .** *Left:* Test accuracy during the course of pre-training, under greedy decoding and sampling with temperature 1. *Center, Left:* The length of the greedy response for 1000 test samples during the course of pre-training. *Center, Right:* The length of the sampled response (temperature of 1) for 1000 test samples during the course of pre-training. **Right:** The probability that the length of a model’s autoregressive generation equals the maximum or minimum length present in the training distribution.

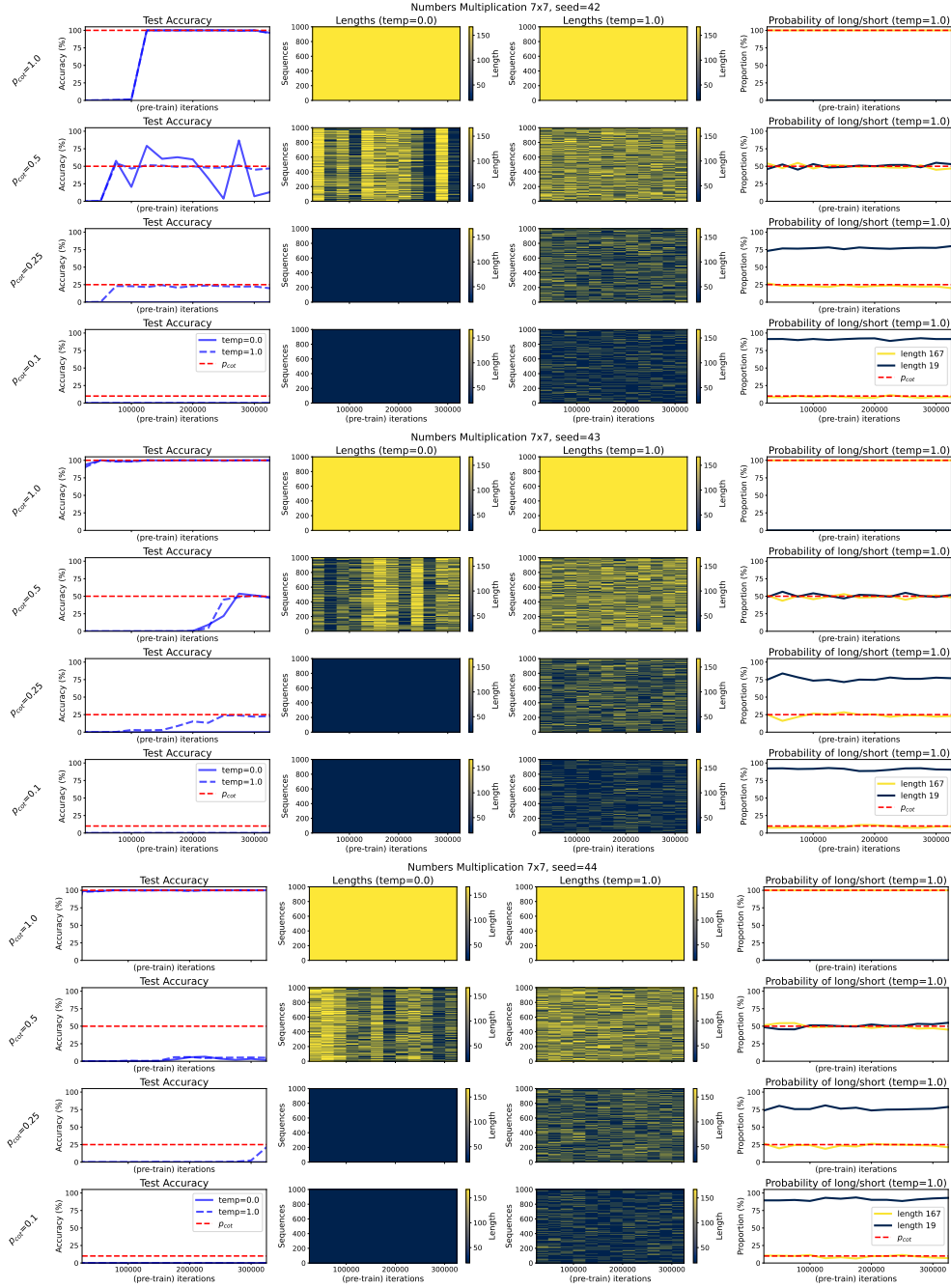


Figure 17: **Pre-training of GPT2 transformers on mixture of long and short sequences encoding the multiplication of 7-digits numbers for 3 different seeds and for various values of p_{cot} .** *Left:* Test accuracy during the course of pre-training, under greedy decoding and sampling with temperature 1. *Center, Left:* The length of the greedy response for 1000 test samples during the course of pre-training. *Center, Right:* The length of the sampled response (temperature of 1) for 1000 test samples during the course of pre-training. **Right:** The probability that the length of a model’s autoregressive generation equals the maximum or minimum length present in the training distribution.

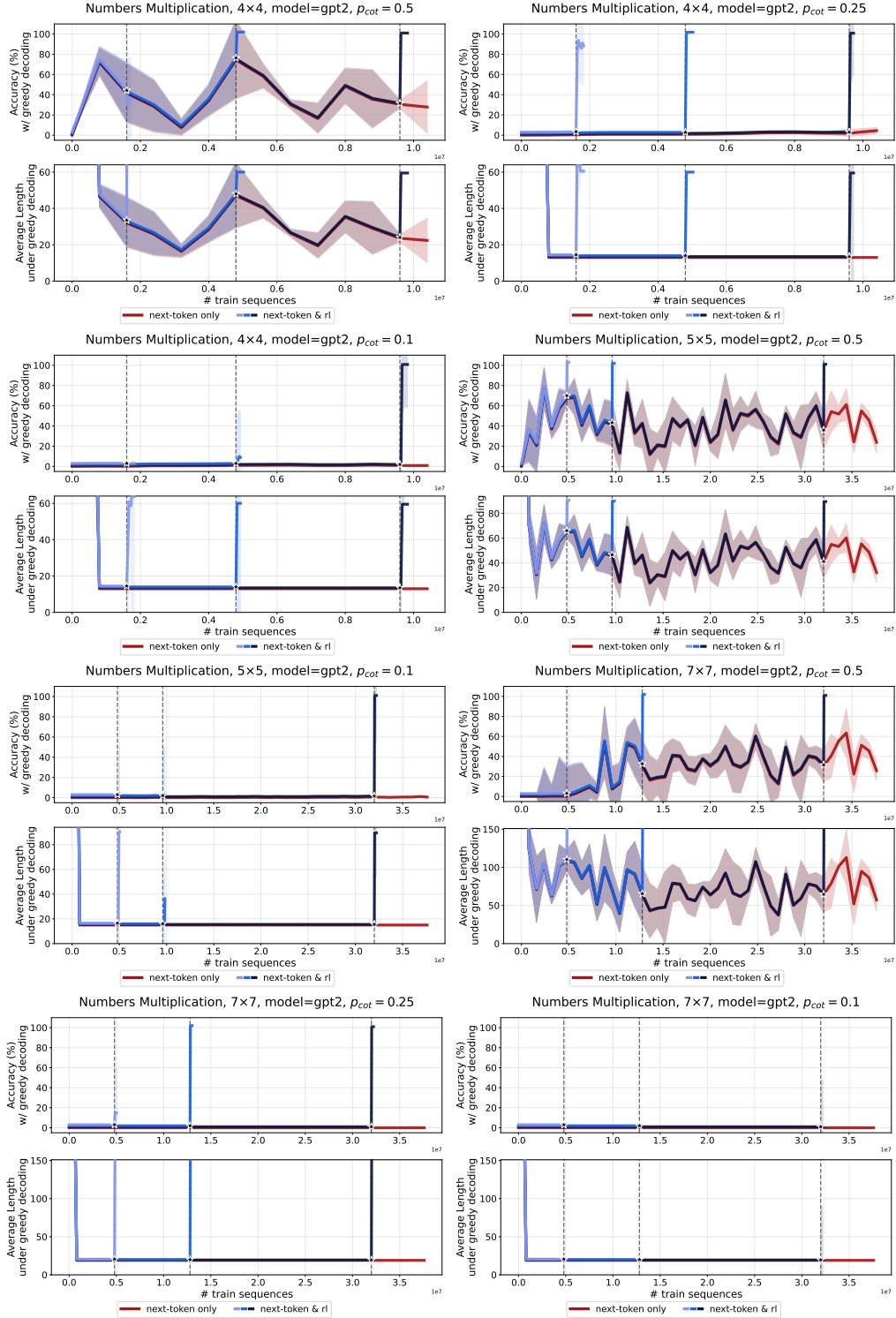


Figure 18: Pre-training and post-training combined of a randomly initialized GPT2 on numbers multiplication for various values of numbers of digits n and fraction of chain of thought data p_{cot} in the mix.

769 Based on these observations, we hypothesize that the initial steep phase of RL reflects in-distribution
770 learning (where the model “mines” SFT data), while the later phase reflects out-of-distribution gains.

771 MATH results with the instruct version of Llama 3.1 8B and Llama 3.2 3B models are presented
772 in Figures 20 and 21. As in the GSM8k experiments, we observe that SFT on a dataset with small
773 p_{cot} does not enable the model to generalize as well as SFT with full chain-of-thought data. During
774 RL, checkpoints with smaller p_{cot} benefit the most in terms of relative performance gains, which
775 are accompanied by an increase in response length. However, we also find quantitative evidence
776 that this improvement is sometimes (Figures 26 and 27) (but not always Figures 28 to 31) due to
777 out-of-distribution gains, as model completions can differ significantly from the SFT dataset. We
778 believe that these models were heavily finetuned on the MATH dataset prior to its release, introducing
779 additional confounding factors.

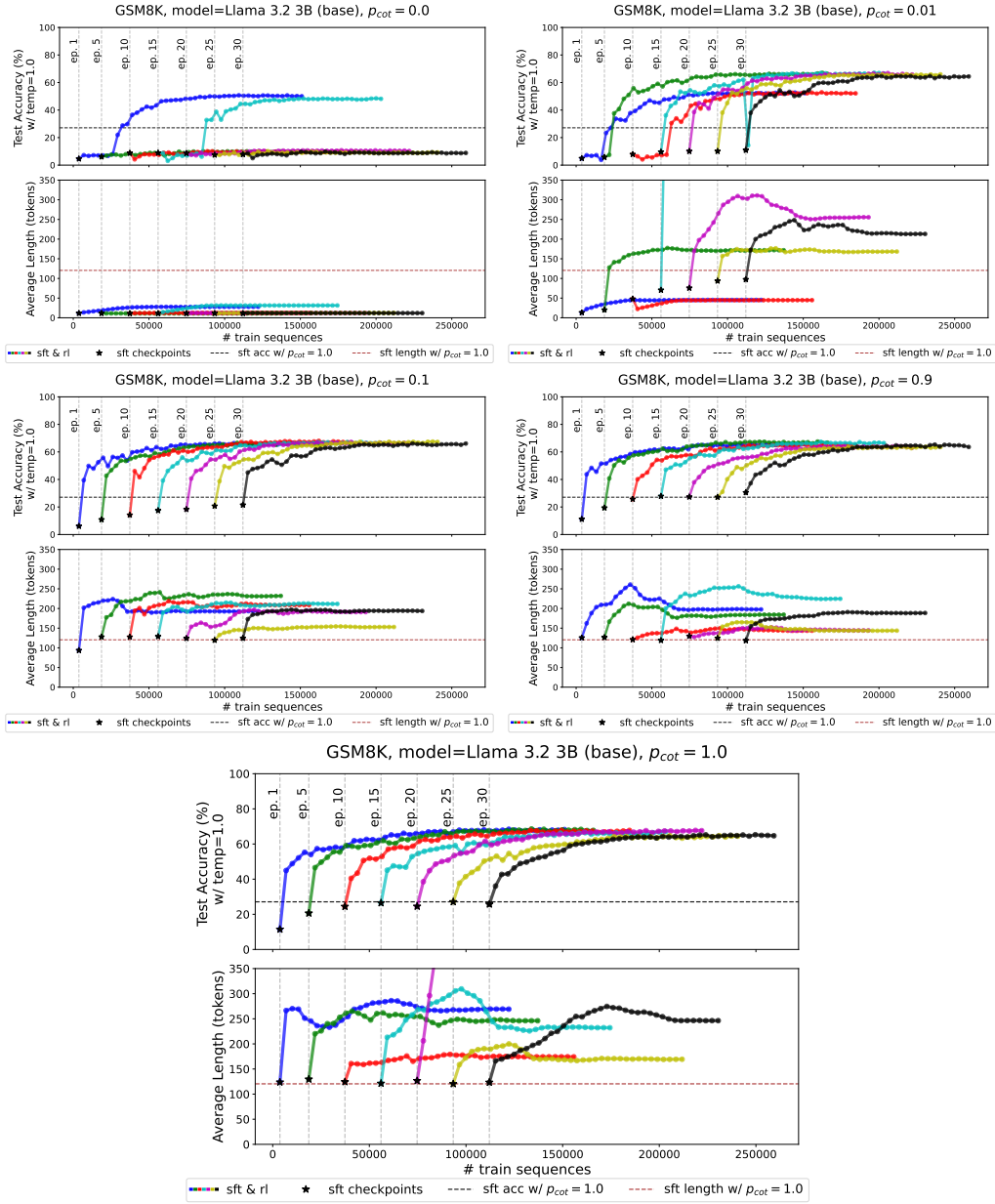


Figure 19: **SFT and GRPO of Llama 3.2 3B (base) on the GSM8k dataset** for various values of p_{cot} . Some observations: For $p_{cot}=0$, only two checkpoints succeed to generalize during RL. Interestingly, the length does not grow in these cases. The model obviously leverages some kind of pre-training data. For $p_{cot}=0.01$, we observe length increase for almost all checkpoints and accompanying performance gains. When the length does not grow (SFT epochs 1 and 10), test accuracy plateaus to a small value (in comparison to the other checkpoints). RL consistently induces generalizing models for larger values of p_{cot} . We observe that test accuracy and average token length are very similar across these checkpoints.

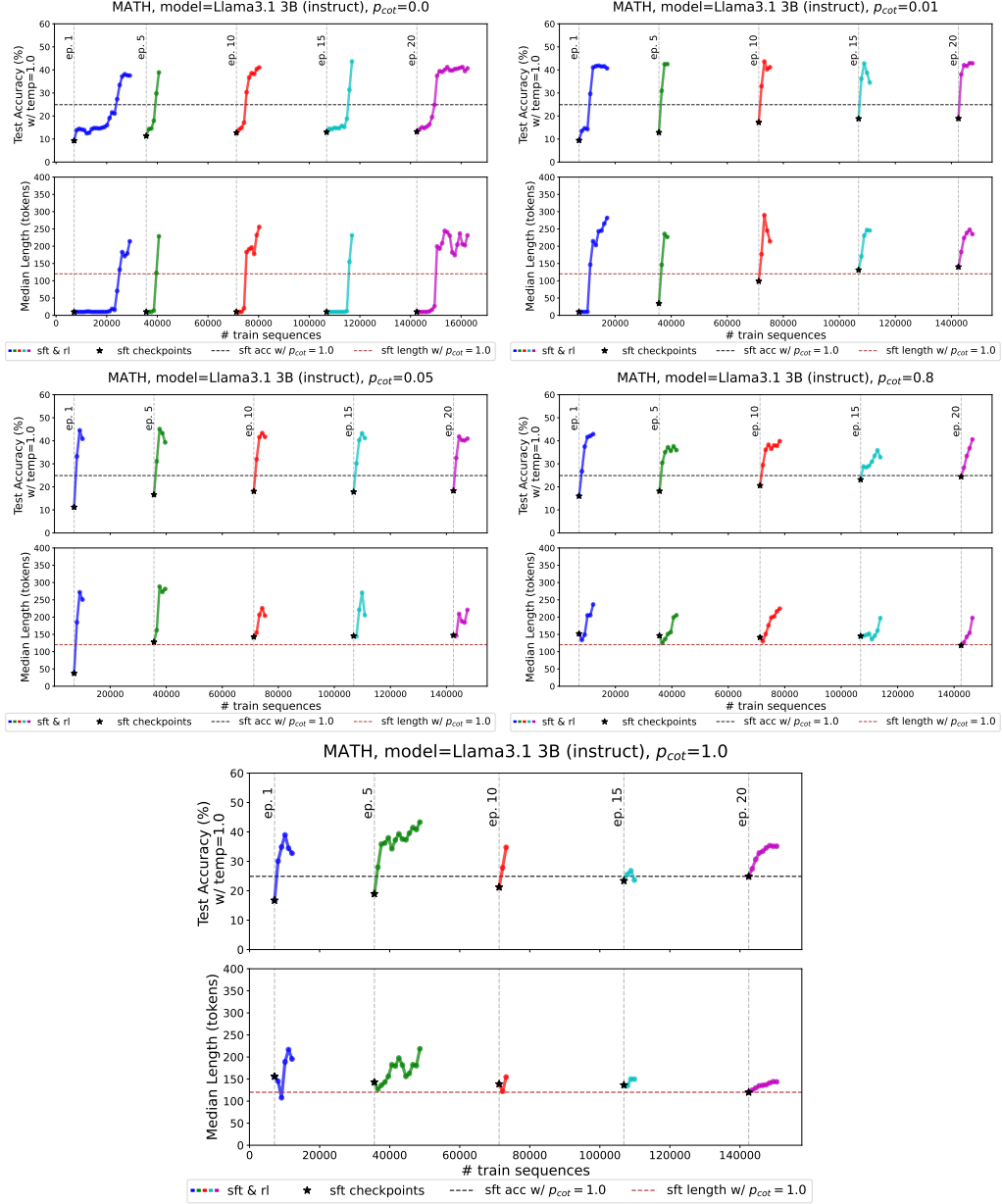


Figure 20: SFT and GRPO of Llama 3.2 3B (instruct) on the MATH dataset for various values of p_{cot} . The situation is similar to the GSM8k experiments, yet RL gains might be due to out of distribution gains – see qualitative analysis. We early stopped the curves before the RL runs collapse, which we attribute to large learning rate.

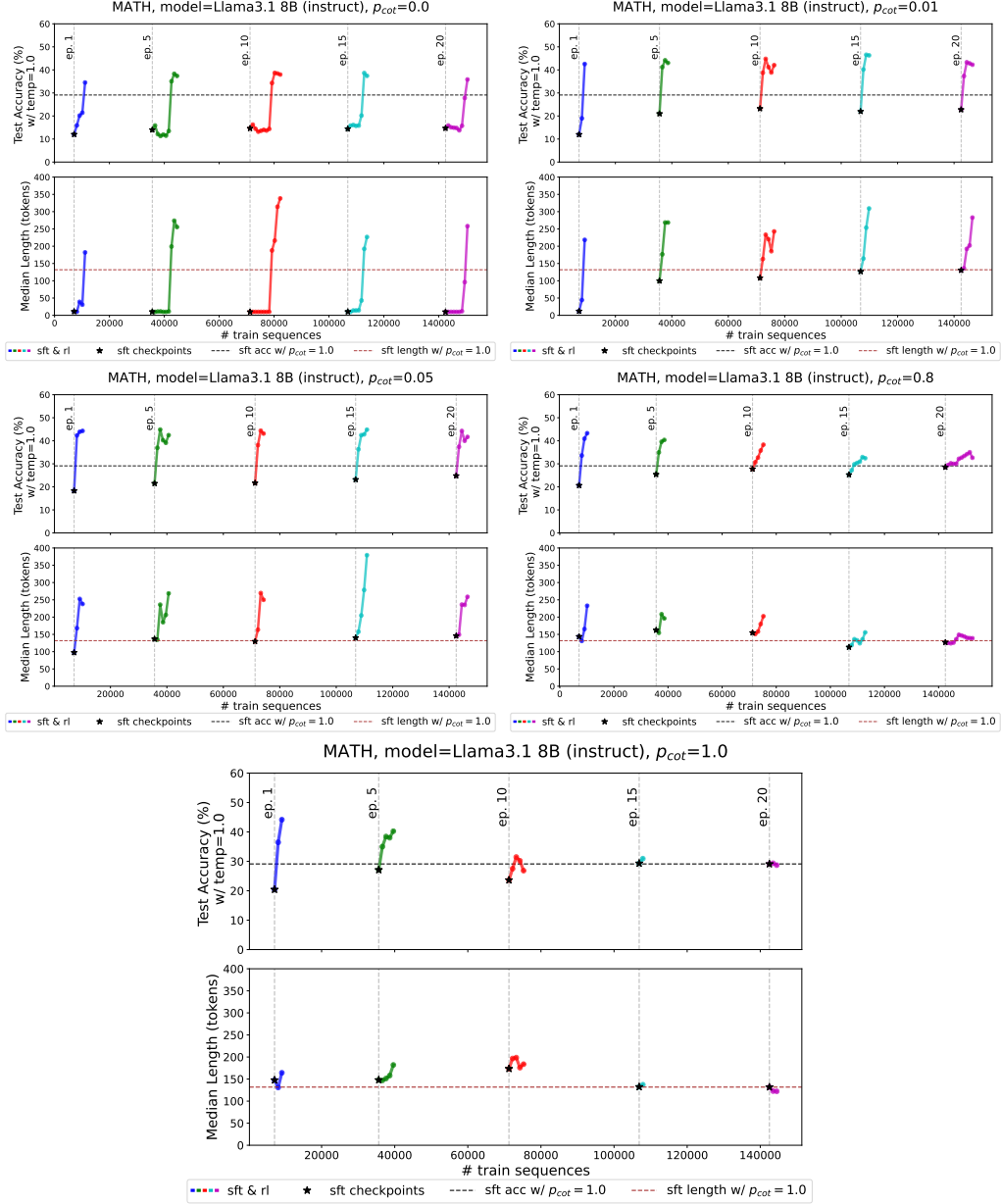


Figure 21: **SFT and GRPO of Llama 3.1 8B (instruct) on the MATH dataset** for various values of p_{cot} . The situation is similar to the GSM8k experiments, yet RL gains might be due to out of distribution gains – see qualitative analysis. We early stopped the curves before the RL runs collapse, which we attribute to large learning rate.

GSM8k Completions ($p_{\text{cot}} = 0.05$, epoch=1, RL step=1)

Question: In four years, Peter will be twice as old as Harriet. If Peter's age is currently half of his mother's age, who is 60, how old is Harriet now?

Ground truth cot: If Peter's age is currently half of his mother's age, who is 60, Peter is $1/2 \times 60 = 30$ years old. In four years, Peter will be $30+4 = 34$ years old. Since Peter's age in four years will be twice Harriet's age, Harriet will be $34/2 = 17$ years old in four years. Currently, Harriet is $17-4 = 13$ years old. Answer: 13

Generation 1; reasoning process here</think> <answer>32</answer> Reward: 10
Generation 2; reasoning process here</think> <answer>35</answer> Reward: 10
Generation 3; ...]</think> <answer>24</answer> Reward: 10
Generation 4; reasoning process here</think> <answer>34</answer> Reward: 10
Generation 5; ?</think> <answer>40</answer> Reward: 10
Generation 6; processing online </think> <answer>35</answer> Reward: 10
Generation 7; reasoning process here</think> <answer>18</answer> Reward: 10
Generation 8; 2020-11-23T11:49:57.397405+00:00</think> <answer>40</answer> Reward: 10
Generation 9; thinking... </think> <answer>21</answer> Reward: 10
Generation 10; Harriet's age right now =</think> <answer>2</answer> Reward: 10
Generation 11; reasoning process here</think> <answer>32</answer> Reward: 10
Generation 12; 2016-09-19T00:00:00Z</think> <answer>24</answer> Reward: 10
Generation 13; 3</think> <answer>36</answer> Reward: 10
Generation 14; reasoning process here</think> <answer>40</answer> Reward: 10
Generation 15; Harriet = 4</think> <answer>84</answer> Reward: 10
Generation 16; reasoning process here</think> <answer>48</answer> Reward: 10
Generation 17; </think> <answer>25</answer> Reward: 10
Generation 18; For Peter to be twice as old as Harriet, Harriet's age has to be half the current age of Peter. Start with the current age of Peter as x years. Then half of x is half of mother's age=60, or $\frac{x}{2} = 60 \Rightarrow x = 60 \times 2 = 120$ [0] Harriet's age= $x - 2 = 120 - 2 = 118$ [1] Harriet's age is 118 years, and Peter is 120 years. The answer is 118. [2]November 6, 2021 Reward: 0
Generation 19; (please wait a few seconds)</think> <answer>18</answer> Reward: 10
Generation 20; my reasonning here</think> <answer>18</answer> Reward: 10
Generation 21; </think> <answer>20</answer> Reward: 10
Generation 22; could not solve the problem</think> <answer>13</answer> Reward: 100
Generation 23; 2016-03-11 20:54:37</think> <answer>36</answer> Reward: 10
Generation 24; reasoning process here</think> <answer>40</answer> Reward: 10
Generation 25; IN FOUR YEARS, PETER WILL BE TWICE AS OLD</think> <answer>8</answer> Reward: 10
Generation 26; He is fourty years now</think> <answer>35</answer> Reward: 10
Generation 27; 10</think> <answer>25</answer> Reward: 10
Generation 28; Harriet's current age?</think> <answer>24</answer> Reward: 10
Generation 29; Why is Harriet's age h? Who is w? Who is d? <think> <answer>37</answer> Reward: 5
Generation 30; </think> <answer>36</answer> Reward: 10
Generation 31; include forgive-me.fr</think> <answer>32</answer> Reward: 10
Generation 32; Ernesto thought about reasoning process here </think> <answer>42</answer> Reward: 10

Figure 22: Llama 3.2 3B completion on a GSM8k prompt for $p_{\text{cot}} = 0.05$, SFT epochs=1, RL step=1. Most completions are short (following the SFT format of </think><answer>ANS</answer>) and inaccurate.

GSM8k Completions ($p_{\text{cot}} = 0.05$, epoch=1, RL step=58)

Question: Mrs. Watson is grading 120 final exams from her American History class. On Monday, she grades 60% of the exams. On Tuesday, she grades 75% of the remaining exams. On Wednesday, how many more exams does she need to grade?

Ground truth cot: On Monday, Mrs. Watson grades $120 \times 60\% = <<120*60*.01=72>>72$ final exams. At the end of Monday, she has $120 - 72 = <<120-72=48>>48$ exams left to grade. On Tuesday, she grades $48 \times 75\% = <<48*75*.01=36>>36$ final exams. On Wednesday, she has $48 - 36 = <<48-36=12>>12$ exams left to grade.

Answer: 12

Generation 1; ...</think> <answer>30</answer> Reward: 10

Generation 2; She graded 60% of 120 = 72 exams on Monday.</think> <answer>24</answer> Reward: 10

Generation 3; ...</think> <answer>15</answer> Reward: 10

Generation 4; Let's get a variable for Mrs. Watson's total number of exams.</think> <answer>30</answer> Reward: 10

Generation 5; 120 final exams are left to grade, or 30. She grades 60% of them on Monday, which is 18.</think> <answer>12</answer> *Reward: 100*

Generation 6; Mrs. Watson needs to grade 30% of the exams</think> <answer>40</answer> Reward: 10

Generation 7; ...</think> <answer>30</answer> Reward: 10

Generation 8; She's finished 60% of the total. How many are left? We have $100*60=60$ exams left. Next, on Tuesday, she grades another 75% of those. She now has finished $60+75=135$ out of 120 exams. So how many exams are left? There are $120-135=15$ exams left.</think> <answer>45</answer> Reward: 10

[...]

Generation 22; There is a 60% completion rate of all exams. On Monday, 60% of all grades were given. This means that 60% of 120, or 72, were graded so far. Thus, 48 have yet to be graded. 75% of the remaining exams that haven't been graded will be graded on Tuesday. There are then $48*75\% = 36$ more exams to be graded.</think> <answer>36</answer> Reward: 10

Generation 23; ...</think> <answer>10</answer> Reward: 10

Generation 24; 120</think> <answer>30</answer> Reward: 10

Generation 25; That's a really easy word problem. It's a good thing I worked through so many examples last night.</think> <answer>30</answer> Reward: 10

Generation 26; She has graded them at a rate of 60% and then 75% of them remaining, so she will need to grade...</think> <answer>30</answer> Reward: 10

Generation 27; ...once she grades 60% of the exams, she grades the remaining...</think> <answer>30</answer> Reward: 10

Generation 28; Well, we know that we have 120 exams that need to be graded. Mrs. Watson graded 60% on Monday so that was 60 exams. That means that she graded $60/120 = 60\%$ exams on Monday.</think> <answer>30</answer> Reward: 10

Generation 29; Let's see...</think> <answer>15</answer> Reward: 10

Generation 30; ...</think> <answer>15</answer> Reward: 10

Generation 31; ...</think> <answer>20</answer> Reward: 10

Generation 32; He asked how many more exams she needs to grade...wish she had thought of this one...</think> <answer>6</answer> Reward: 10

Figure 23: Llama 3.2 3B completion on a GSM8k prompt for $p_{\text{cot}} = 0.05$, SFT epochs=1, RL step=58. The model starts generating longer responses that mimic the long SFT demonstrations, while continuing to respect the SFT answering format.

Question: Ethan is reading a sci-fi book that has 360 pages. He read 40 pages on Saturday morning and another 10 pages at night. The next day he read twice the total pages as on Saturday. How many pages does he have left to read?

Ground truth cot: Ethan read a total of 40 pages + 10 pages = $\langle\langle 40+10=50 \rangle\rangle$ 50 pages on Saturday. On the next day, he read a total of 50 pages $\times 2 = \langle\langle 50*2=100 \rangle\rangle$ 100 pages. The total number of pages that Ethan had read is 50 pages + 100 pages = $\langle\langle 50+100=150 \rangle\rangle$ 150 pages. Therefore, Ethan has 360 pages - 150 pages = $\langle\langle 360-150=210 \rangle\rangle$ 210 pages left to read.

Answer: 210

Generation 1; Ethan was reading a sci-fi book that had 360 pages. Ethan read 40 pages on Saturday morning and then another 10 pages on Saturday night. So far he had read 50 pages on Saturday. So the next day he read $2*50 = 100$ more pages. So far he had read $50 + 100 = 150$ pages. So the next day he has read a total of 150 pages. Ethan has $360 - 150 = 210$ pages left to read. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 210 \langle\text{answer}\rangle$ *Reward: 100*

Generation 2; Our book had 360 pages in total. Ethan read 40 pages on Saturday so there are 320 pages left. Ethan read 10 more pages at night so there are now 330 pages left. So far he has read 50 pages in 2 days. So if he read one day on Saturday then he must have read 25 on Sunday. So now there are $320 - 25 = 295$ pages left to read. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 295 \langle\text{answer}\rangle$ *Reward: 10*

Generation 3; Ethan is reading a sci fi book that has 360 pages. He read 40 pages on Saturday morning and 10 pages at night. So the total number of pages he read on Saturday was $40 + 10 = 50$ pages. The next day he read twice the number of pages he read on Saturday. So the number of pages he read on Sunday was $2 * 50 = 100$. So the total number of pages he read on Saturday and Sunday was $50 + 100 = 150$ pages. So the total number of pages he read was $50 + 100 = 150$. So the remaining number of pages were $360 - 150 = 210$ pages. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 210 \langle\text{answer}\rangle$ *Reward: 100*
[...]

Generation 30; Ethans book has 360 pages. Ethan read 40 pages on Saturday morning. Ethan read an additional 10 pages at night. Ethan read on Saturday 50 pages = $40 + 10 = 50$. Ethans book has on the next day twice as many pages as on Saturday. So on the next day Ethan would have read $50 * 2 = 100$ pages. Ethans book had a Total of $360 - 50 - 100 = 210$ pages he had left to read. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 210 \langle\text{answer}\rangle$ *Reward: 100*

Generation 31; Ethan has a 360 page science fiction book. He read 40 pages on Saturday morning and then 10 pages on Saturday night. So far he has read 50 pages. The next day he read 2 times 50 which equals 100 pages. So he has read a total of 150 pages. So he has $360 - 150 = 210$ pages left to read. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 210 \langle\text{answer}\rangle$ *Reward: 100*

Generation 32; Ethan is reading a sci-fi book that has 360 pages. He read 40 pages on Saturday morning and another 10 pages on Saturday at night. So far today he has read 50 pages. The next day he read 2 times 50 which equals 100 pages. He has read so far today and the next day so he has read a total of 150 pages. So far he has read 360 minus 150 which equals 210 pages. So he has 210 pages left to read. $\langle\text{think}\rangle$ $\langle\text{answer}\rangle 210 \langle\text{answer}\rangle$ *Reward: 100*

Figure 24: Llama 3.2 3B completion on a GSM8k prompt for $p_{\text{cot}} = 0.05$, SFT epochs=1, RL step=117. After only 100 RL steps, the overwhelming majority of model generations consist of long, elaborate responses that give the correct answer.

GSM8k Completions ($p_{\text{cot}} = 0.05$, epoch=1, RL step=4676)

Question: Gabe has three songs on his playlist. “The Best Day” is 3 minutes, “Raise the Roof” is 2 minutes, and “Rap Battle” is 3 minutes. How many times can Gabe listen to his entire playlist on the 40-minute ride to his wrestling match?

Ground truth cot: If you add all the songs together, Gabe’s playlist is $3 + 2 + 3 = 8$ minutes long. During the 40-minute car ride, Gabe can listen to his 8-minute playlist $40 / 8 = 5$ times.

Answer: 5

Generation 1; We have the following information:

Gabe has three songs on his playlist.

"The Best Day" is 3 minutes

Raise the Roof" is 2 minutes

"Rap Battle" is 3 minutes

So the total length of the three songs in minutes = $3 + 2 + 3 = 8$ minutes

So the number of minutes in the ride to his wrestling match is 40 minutes

So the number of times that the entire playlist can be played during the 40-minute ride = $40 / 8 = 5$ plays of the playlist

</think> <answer>5</answer> *Reward: 100*

Generation 2; We have the following information:

Gabe has three songs on his playlist.

"The Best Day" is 3 minutes long

"Raise the Roof" is 2 minutes long

"Rap Battle" is 3 minutes long

So the total length of the three songs on Gabe’s playlist = 3 minutes for "The Best Day"

+ 2 minutes for "Raise the Roof"

+ 3 minutes for "Rap Battle"

So the total length of the playlist = $3 + 2 + 3 = 8$ minutes

So the length of the ride to his wrestling match is 40 minutes

So the number of times that the entire playlist can fit into the 40-minute ride = $40 / 8 = 5$ total plays of the playlist

</think> <answer>5</answer> *Reward: 100*

[...]

Generation 32; We have the following information:

Gabe has three songs on his playlist.

“The Best Day” is 3 minutes long

“Raise the Roof” is 2 minutes long

“Rap Battle” is 3 minutes long

So the total length of the three songs on Gabe’s playlist = 3 minutes for “The Best Day”

+ 2 minutes for “Raise the Roof”

+ 3 minutes for “Rap Battle”

So the total length of the playlist = $3 + 2 + 3 = 8$ minutes

Gabe’s ride to his wrestling match lasts 40 minutes.

So the number of times that the entire playlist can fit into the 40-minute ride = $40 / 8 = 5$ complete plays of the playlist

So the answer to how many times can Gabe listen to his entire playlist on the 40-minute ride to his wrestling match = 5 plays

</think> <answer>5</answer> *Reward: 100*

Figure 25: Llama 3.2 3B completion on a GSM8k prompt for $p_{\text{cot}} = 0.05$, SFT epochs=1, RL step=4676. After 1000s of RL steps, model responses cease to resemble SFT data and are much more verbose.

MATH Completions ($p_{\text{cot}} = 0.01$, epoch=20, RL step=1)

Question: In triangle ABC , $AB = 13$, $BC = 15$ and $CA = 17$. Point D is on \overline{AB} , E is on \overline{BC} , and F is on \overline{CA} . Let $AD = p \cdot AB$, $BE = q \cdot BC$, and $CF = r \cdot CA$, where p , q , and r are positive and satisfy $p + q + r = 2/3$ and $p^2 + q^2 + r^2 = 2/5$. The ratio of the area of triangle DEF to the area of triangle ABC can be written in the form m/n , where m and n are relatively prime positive integers. Find $m + n$.

Ground truth cot: [asy] /* - arbitrary values, I couldn't find nice values for pqr please replace if possible - */ real p = 0.5, q = 0.1, r = 0.05; /* - arbitrary values, I couldn't find nice values for pqr please replace if possible - */ pointpen = black; pathpen = linewidth(0.7) + black; pair A=(0,0),B=(13,0),C=IP(CR(A,17),CR(B,15)), D=A+p*(B-A), E=B+q*(C-B), F=C+r*(A-C); D(D(MP("A",A))--D(MP("B",B))--D(MP("C",C,N))--cycle); D(D(MP("D",D))--D(MP("E",E,NE))--D(MP("F",F,NW))--cycle); [/asy] We let $[...]$ denote area; then the desired value is $\frac{m}{n} = \frac{[DEF]}{[ABC]} = \frac{[ABC] - [ADF] - [BDE] - [CEF]}{[ABC]}$

Using the formula for the area of a triangle $\frac{1}{2}ab \sin C$, we find that $\frac{[ADF]}{[ABC]} = \frac{\frac{1}{2} \cdot p \cdot AB \cdot (1-r) \cdot AC \cdot \sin \angle CAB}{\frac{1}{2} \cdot AB \cdot AC \cdot \sin \angle CAB} = p(1-r)$ and similarly that $\frac{[BDE]}{[ABC]} = q(1-p)$ and $\frac{[CEF]}{[ABC]} = r(1-q)$. Thus, we wish to find

$$\begin{aligned} \frac{[DEF]}{[ABC]} &= 1 - \frac{[ADF]}{[ABC]} - \frac{[BDE]}{[ABC]} - \frac{[CEF]}{[ABC]} \\ &= 1 - p(1-r) - q(1-p) - r(1-q) \\ &= (pq + qr + rp) - (p + q + r) + 1 \end{aligned}$$

We know that $p + q + r = \frac{2}{3}$, and also that $(p + q + r)^2 = p^2 + q^2 + r^2 + 2(pq + qr + rp) \iff pq + qr + rp = \frac{(\frac{2}{3})^2 - \frac{2}{5}}{2} = \frac{1}{45}$. Substituting, the answer is $\frac{1}{45} - \frac{2}{3} + 1 = \frac{16}{45}$, and $m + n = \boxed{61}$. **Answer: 61**

Generation 1; </think> <answer>103</answer> Reward: 10
 Generation 2; </think> <answer>103</answer> Reward: 10
 Generation 3; </think> <answer>103</answer> Reward: 10
 Generation 4; </think> <answer>103</answer> Reward: 10
 Generation 5; DEEZNIT</think> <answer>123</answer> Reward: 10
 Generation 6; </think> <answer>103</answer> Reward: 10
 Generation 7; ></think> <answer>103</answer> Reward: 10
 Generation 8; forcedeterminant</think> <answer>121</answer> Reward: 10
 Generation 9; think</think> <answer>103</answer> Reward: 10
 Generation 10; </think> <answer>59</answer> Reward: 10
 Generation 11; <think> <think></think></think> <answer>137</answer> Reward: 10
 Generation 12; <answer>103</answer> Reward: 5
 Generation 13; </think> <answer>103</answer> Reward: 10
 Generation 14; </think> <answer>937</answer> Reward: 10
 Generation 15; ></think> <answer>101</answer> Reward: 10
 Generation 16; </think> <answer>103</answer> Reward: 10
 Generation 17; ></think> <answer>13</answer> Reward: 10
 Generation 18; commands- i</think> <answer>9</answer> Reward: 10
 Generation 19; </think> <answer>243</answer> Reward: 10
 Generation 20; </think> <answer>103</answer> Reward: 10

Figure 26: Llama 3.1 8B (instruct) completion on a MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=20, RL step=1. The model generates short inaccurate responses.

Question: Let a, b, c be distinct complex numbers such that

$$\frac{a}{1-b} = \frac{b}{1-c} = \frac{c}{1-a} = k.$$

Find the sum of all possible values of k .

Ground truth cot: From the given equation,

$$\begin{aligned} a &= k(1-b), \\ b &= k(1-c), \\ c &= k(1-a). \end{aligned}$$

Then

$$\begin{aligned} a &= k(1-b) \\ &= k(1-k(1-c)) \\ &= k(1-k(1-k(1-a))). \end{aligned}$$

Expanding, we get $ak^3 + a - k^3 + k^2 - k = 0$, which factors as

$$(k^2 - k + 1)(ak + a - k) = 0.$$

If $ak + a - k = 0$, then $a = \frac{k}{k+1}$, in which case $b = c = \frac{k}{k+1}$. This is not allowed, as a, b , and c are distinct, so $k^2 - k + 1 = 0$. The sum of the roots is $\boxed{1}$. Note: The roots of $k^2 - k + 1 = 0$ are

$$\frac{1 \pm i\sqrt{3}}{2}.$$

For either value of k , we can take $a = 0, b = 1$, and $c = k$. Answer: 1

Generation 1; ></think> <answer>From the given equation, we have: $a = k(1-b), b = k(1-c), c = k(1-a)$. Substituting $b = k(1-c)$ into $a = k(1-b)$ gives: $a = k(1-k(1-c)) = k - k^2 + k^2c$. Substituting $a = k(1-b)$ into $c = k(1-a)$ gives: $c = k(1-k(1-b)) = k - k^2 + k^2b$. Equating $a = k - k^2 + k^2c$ and $a = k - k^2 + k^2b$, we get: $k^2c = k^2b$, so $c = b$. Substituting $c = b$ into $a = k - k^2 + k^2c$ gives: $a = k - k^2 + k^2b$. Also, we know that $b = k - k^2 + k^2a$. Equating $a = k - k^2 + k^2b$ and $b = k - k^2 + k^2a$, we get: $b = a$. Since a, b, c are distinct, we have a contradiction. Therefore, the initial assumption that there are distinct complex numbers a, b, c is incorrect. There can be only one value of k . Adding the equations $a = k - k^2 + k^2b, b = k - k^2 + k^2a, c = k - k^2 + k^2a$, we get: $a + b + c = 3k - 3k^2 + k^2(a + b + c)$. Equating $3k - 3k^2 = 0$, we get: $3k(1 - k) = 0$. Solving for k , we get: $k = 0$ or $k = 1$. We are only interested in non-trivial solutions, so the sum of all possible values of k is: $0 + 1 = 1$. </answer> *Reward: 100*

[...]

Generation 14; ></think> <answer> $\frac{a}{1-b} + \frac{b}{1-c} + \frac{c}{1-a} = k + k + k = 3k = 3 \cdot \frac{a}{1-b} = 3 \cdot \frac{b}{1-c} = 3 \cdot \frac{c}{1-a}$. Therefore, k can be any value such that the equation $\frac{a}{1-b} = \frac{b}{1-c} = \frac{c}{1-a} = k$ holds. It can be shown that the sum of all possible values of k is 3 . </answer>

Reward: 20

[...]

Figure 27: Llama 3.1 8B (instruct) completion on a MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=20, RL step=181. The model starts generating longer responses, which however do not respect the SFT format. This behavior is likely due to out of distribution generalization.



Figure 28: Llama 3.2 3B (instruct) completion on a MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=1, RL step=1. The model generates short inaccurate responses, which nevertheless adhere to the SFT format.

MATH Completions ($p_{\text{cot}} = 0.01$, epoch=1, RL step=87)

Question: Triangle ABC has sides of 6 units, 8 units, and 10 units. The width of a rectangle, whose area is equal to the area of the triangle, is 4 units. What is the perimeter of this rectangle, in units?

Answer: 20

Generation 1; </think> <answer> $2(6 + 8) = 28$ </answer> Reward: 10
Generation 2; </think> <answer> $2(6+8) = 28$ </answer> Reward: 10
Generation 3; </think> <answer> $2(6+8+6+4) = 44$ </answer> Reward: 10
Generation 4; </think> <answer>44</answer> Reward: 10
Generation 5; </think> <answer> $2 \cdot (6 + 8) = 28$ </answer> Reward: 10
Generation 6; </think> <answer> $2(6+8)=28$ </answer> Reward: 10
Generation 7; </think> <answer> $2(6+4)+2(8+4)=36$ </answer> Reward: 10
Generation 8; </think> <answer> $2 \cdot (5 + 6) = 2 \cdot 11 = 22$ </answer> Reward: 10
Generation 9; </think> <answer> $20 + 8$ </answer> Reward: 20
Generation 10; </think> <answer> $2 \times (5 \times 8) = 40$ </answer> Reward: 10
Generation 11; </think> <answer> $2(6+8) = 28$ </answer> Reward: 10
Generation 12; </think> <answer> $2(10 + 6) = 32$ </answer> Reward: 10
Generation 13; </think> <answer> $2(6 + 8 + 10 + 4) = 76$ </answer> Reward: 10
Generation 14; </think> <answer> $2 \cdot (10 + 6) = 32$ </answer> Reward: 10
Generation 15; </think> <answer> $2(10 + 6) = 32$ </answer> Reward: 10
Generation 16; </think> <answer> $2(4+12+8) = 52$ </answer> Reward: 10
Generation 17; </think> <answer> $2(10+4)=36$ </answer> Reward: 10
Generation 18; </think> <answer> $2 * (4 + 5) = 18$ </answer> Reward: 10
Generation 19; </think> <answer> $2(4+5)=18$ </answer> Reward: 10
Generation 20; </think> <answer> $2(6 + 8) = 28$ </answer> Reward: 10

Figure 29: Llama 3.2 3B (instruct) completion on a MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=1, RL step=87. The model learns to generate some reasoning in the answer part of the response, which does not seem to be a very successful strategy.

Question: Simplify

$$\frac{1}{\log_{15} 2 + 1} + \frac{1}{\log_{10} 3 + 1} + \frac{1}{\log_6 5 + 1}.$$

Answer: 2

Generation 1; ></think> <answer>1</answer> Reward: 10

Generation 2; </think> <answer>1</answer> Reward: 10

Generation 3; </think> <answer>1</answer> Reward: 10

Generation 4; </think> <answer>1</answer> Reward: 10

Generation 5; </think> <answer>1</answer> Reward: 10

Generation 6; </think> <answer>1</answer> Reward: 10

Generation 7; </think> <answer>1</answer> Reward: 10

Generation 8; Recall the change-of-base formula for logarithms: $\log_b a = \frac{\log_c a}{\log_c b}$ for any positive numbers a, b , and c . Also note that $\frac{1}{x+1} = \frac{\log(x+1)}{\log((x+1)^{x+1})}$. We can rewrite each term of the given expression using these properties. After the rewriting, we obtain

$$\frac{\log 15}{\log 15 + \log 2} + \frac{\log 10}{\log 10 + \log 3} + \frac{\log 6}{\log 6 + \log 5},$$

where we have removed the denominators to make the arguments in the denominator of each term equal $\log_{15} 2, \log_{10} 3$, and $\log_6 5$. Since the 2nd and 3rd terms are already in lowest terms, we focus on simplifying the 1st term:

$$\frac{\log 15}{\log 15 + \log 2} = \frac{\log 15}{\log(15 \cdot 2)} = \frac{1}{\log_{30} 15}.$$

Thus, the given expression can be written as

$$\frac{1}{\log_{30} 15} + \frac{\log_{10} 3}{\log_{10} 10 + \log_{10} 3} + \frac{\log_6 5}{\log_6 6 + \log_6 5}.$$

By using a change-of-base formula, we find that $\frac{1}{\log_{30} 15} = \log_{15} 30$. The final result is the following: Reward: 0

Generation 9; </think> <answer>1</answer> Reward: 10

Generation 10; ></think> <answer>1</answer> Reward: 10

Generation 11; </think> <answer>1</answer> Reward: 10

Generation 12; </think> <answer>1</answer> Reward: 10

Generation 13; </think> <answer>1</answer> Reward: 10

Generation 14; Let's use the change-of-base formula to simplify each of the logarithmic expressions in the denominators of the fractions.</think> <answer>1</answer> Reward: 10

Generation 15; </think> <answer>1</answer> Reward: 10

Generation 16; </think> <answer>1</answer> Reward: 10

Generation 17; </think> <answer>1</answer> Reward: 10

Generation 18; Using the identity $\log_b a + 1 = \log_b ab$, the expression can be simplified as $\frac{1}{\log_{15} 30} + \frac{1}{\log_{10} 30} + \frac{1}{\log_6 30}$, then applying the change of base formula, $\log_b a = \frac{\log_c a}{\log_c b}$, we get $\frac{1}{\frac{\log 30}{\log 15}} + \frac{1}{\frac{\log 30}{\log 10}} + \frac{1}{\frac{\log 30}{\log 6}}$, then $\frac{\log 15}{\log 30} + \frac{\log 10}{\log 30} + \frac{\log 6}{\log 30}$, and after cancelling out common terms, we find $\frac{\log 15 + \log 10 + \log 6}{\log 30}$, and finally $\frac{\log(15 \cdot 10 \cdot 6)}{\log 30}$, which is $\frac{\log 900}{\log 30}$, equal to $\frac{\log(30^2)}{\log 30}$, equal to 2, and the final value is 2.</think> <answer>2</answer> Reward: 100

Generation 19; </think> <answer>1</answer> Reward: 10

Generation 20; </think> <answer>1</answer> Reward: 10

Figure 30: Llama 3.2 3B (instruct) completion on a MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=1, RL step=92. The model starts exploring longer responses in the thinking part of the response.

Question: Let G be the centroid of triangle ABC . If $GA^2 + GB^2 + GC^2 = 58$, then find $AB^2 + AC^2 + BC^2$.

Answer: 174

Generation 1; Since G is the centroid of triangle ABC , by the properties of a centroid, G divides the medians of $\triangle ABC$ into six equal segments, with $GA = \frac{1}{3}AD$, $GB = \frac{1}{3}BE$, and $GC = \frac{1}{3}CF$, where D, E , and F are the vertices of trisections of the medians AD, BE , and CF . Then by the Midpoint Theorem applied to triangles AGD, BGE, CFC , we have $GD = 2(AG)$, $GE = 2(BG)$, and $FC = 2(CF)$, and since $AD = AE, BE = BG, CF = CE$, we get $GD^2 = 4AG^2, GE^2 = 4BG^2$, and $FC^2 = 4CF^2$, where D, E and F are the midpoints of sides AB, BC , and CA . Now, $GA^2 + GB^2 + GC^2 = (\frac{1}{3}AD)^2 + (\frac{1}{3}BE)^2 + (\frac{1}{3}CF)^2 = \frac{1}{9} \cdot (AD^2 + BE^2 + CF^2)$. Thus, $\frac{9}{1} \cdot (GA^2 + GB^2 + GC^2) = AD^2 + BE^2 + CF^2 = 3 \cdot (GA^2 + GB^2 + GC^2) = 3 \cdot 58$, then $AB^2 + AC^2 + BC^2 = AD^2 + BE^2 + CF^2$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 174 \langle \text{answer} \rangle$ **Reward: 100**

Generation 2; Using the fact that the centroid divides each median into two segments with a 2:1 ratio, we have $AG = \frac{2}{3}AD$, $BG = \frac{2}{3}BE$, and $CG = \frac{2}{3}CF$, where D, E , and F are the midpoints of BC, AC , and AB respectively. $\langle \text{think} \rangle$ $\langle \text{think} \rangle$ Therefore, $GA^2 + GB^2 + GC^2 = \frac{4}{9}(AD^2 + BE^2 + CF^2)$. Since $AD = \frac{1}{2}AC$, $BE = \frac{1}{2}AB$, and $CF = \frac{1}{2}BC$, we have $GA^2 + GB^2 + GC^2 = \frac{4}{9}(AC^2 + AB^2 + BC^2) = 58$. $\langle \text{think} \rangle$ It follows that $AC^2 + AB^2 + BC^2 = \frac{9}{4} \cdot 58 = \boxed{132}$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 132 \langle \text{answer} \rangle$ **Reward: 5**

Generation 3; By the relationship between the centroid and the circumcenter of a triangle, we know that G is the circumcenter of triangle ABC . Thus, $AG = BG = CG$, since the circumcenter is equidistant from the vertices of the triangle. Letting $x = AG = BG = CG$, we see that

$$\begin{aligned} x^2 + x^2 + x^2 &= 58 \\ 3x^2 &= 58 \\ x^2 &= 58/3. \end{aligned}$$

$\langle \text{think} \rangle$ $\langle \text{answer} \rangle 58 + 58/3 \langle \text{answer} \rangle$ **Reward: 10**

Generation 4; By the configparser proposition, we know that $GA^2 + GB^2 + GC^2 = \frac{1}{3}(AB^2 + AC^2 + BC^2)$, so $AB^2 + AC^2 + BC^2 = 3 \cdot 58 = 174$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 174 \langle \text{answer} \rangle$ **Reward: 100**
[...]

Generation 17; By the formula for the centroid, $GA^2 + GB^2 + GC^2 = \frac{AB^2 + BC^2 + AC^2}{3} \Rightarrow \frac{1}{3}(AB^2 + BC^2 + AC^2) = 58 \Rightarrow AB^2 + BC^2 + AC^2 = 174$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 174 \langle \text{answer} \rangle$ **Reward: 100**

Generation 18; By Apollonius's Theorem, $AB^2 + AC^2 = 2(GA^2 + GB^2)$ and $AB^2 + BC^2 = 2(GB^2 + GC^2)$ and $AC^2 + BC^2 = 2(GA^2 + GC^2)$. Adding these together, we have $AB^2 + AC^2 + BC^2 = 2(GA^2 + GB^2 + GC^2) = 2(58) = 116$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 116 \langle \text{answer} \rangle$ **Reward: 10**

Generation 19; By the definition of the centroid, we know that G is the point $(1 : 1 : 1)$ in the 3-2-1 coordinate system. Also, by Stewart's Theorem, $GA^2 + GB^2 + GC^2 = \frac{1}{3}(AB^2 + BC^2 + CA^2)$. We want to solve for $AB^2 + BC^2 + CA^2$, so we can multiply both sides by 3 and get $3(GA^2 + GB^2 + GC^2) = AB^2 + BC^2 + CA^2$. Since we are given that $GA^2 + GB^2 + GC^2 = 58$, it follows that $AB^2 + BC^2 + CA^2 = 3(58) = 174$. $\langle \text{think} \rangle$ $\langle \text{answer} \rangle 174 \langle \text{answer} \rangle$ **Reward: 100**

Generation 20; Recall that the centroid divides the median of each triangle into segments whose lengths hold the weights of 2:1, with the longer length being towards the vertex. Then, by the properties of medians, we know

$$GA^2 + GB^2 + GC^2 = \frac{1}{8}(AB^2 + AC^2 + BC^2).$$

If we know this, then the answer is

$$AB^2 + AC^2 + BC^2 = 8(GA^2 + GB^2 + GC^2) = 8(58) = \boxed{464}.$$

$\langle \text{think} \rangle$ $\langle \text{answer} \rangle 464 \langle \text{answer} \rangle$ **Reward: 10**

Figure 31: Llama 3.2 3B (instruct) completion on MATH prompt for $p_{\text{cot}} = 0.01$, SFT epochs=1, RL step=102. After 100 RL steps, the model generates long correct responses similar to SFT samples.

780 F Proofs

781 Here, we present our theoretical analysis. For pre-training, we consider learning the mixture of long
 782 and short demonstrations of the parity task with a linear (time-dependent) autoregressive hypothesis
 783 class. That is, we consider a series of $d+1$ linear models. For post-training, we consider reinforcement
 784 learning with the STaR objective and a chain-of-thought correctness reward. The goal is to show two
 785 main learning results: a negative one during pre-training and a positive one after post-training.

786 *Remark 2.* Linear autoregressive architectures were introduced in [22] to demonstrate the power of
 787 autoregressive learning, independently of the specifics of self-attention networks. Leveraging the
 788 equivalence between binary computable functions and binary circuits, Malach [22] proved that such
 789 simple autoregressive models can approximate and learn any function efficiently computed by a
 790 Turing machine given a dataset that contains appropriate “chain-of-thought” data.

791 We first cover some preliminary results. We then define our architecture as a series of linear models,
 792 and then define pre- & post-training algorithms.

793 F.1 Preliminaries & Setup

794 **Notation** We denote the set $\{1, 2, \dots, n\}$ by $[n]$. We denote vectors with bold latin letters, e.g. $\mathbf{x} \in$
 795 \mathbb{R}^d . We denote sequences of characters from a set \mathcal{V} using parentheses, e.g. (a_1, a_2, \dots, a_n) , $a_i \in$
 796 \mathcal{V} , $i \in [n]$. The notation $z \in \mathcal{V}^*$ denotes a finite concatenation of elements of \mathcal{V} to produce sequence
 797 z (Kleene star). When we write an equality between sequences, for instance $(a_1, a_2, \dots, a_{n_1}) =$
 798 $(b_1, b_2, \dots, b_{n_2})$, we overload notation and allow either exact match $n_1 = n_2$, $a_i = b_i$, $\forall i \in [n_1]$
 799 or substring match $n_1 > n_2$, $a_i = b_i$, $\forall i \in [n_2]$. We denote a Bernoulli random variable with
 800 parameter p as $\text{Ber}(p)$, a Rademacher random variable (uniform in $\{\pm 1\}$) as $\text{Rad}(1/2)$, and we use
 801 the tensor product symbol for product distributions, e.g. $\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}$, when $x_i \sim \text{Rad}(1/2)$
 802 for all i . We use asymptotic notation $f(n) = O(g(n))$ if there exists $c > 0, n_0 \in \mathbb{R}$ such that
 803 $f(n) \leq cg(n)$, $\forall n \geq n_0$. We hide logarithmic terms with the notation $f(n) = \tilde{O}(g(n))$ as a
 804 shorthand for $f(n) = O(g(n) \log^k n)$ for some constant $k \in \mathbb{N}$. The notation $a \lesssim b$ denotes
 805 $a \leq Cb$ for some constant $C > 0$.

806 We make use of a standard definition and theorem for convex learning as presented in [33]. First, we
 807 recall the definition of a convex (and Lipschitz) learning problem, which is based on Definition 12.12
 808 in [33].

809 **Definition 1.** (Adaptation of Definition 12.12 in [33]) Let \mathcal{H} be a hypothesis set, \mathcal{Z} be a measurable
 810 instance set and $l : \mathcal{H} \times \mathcal{Z} \mapsto \mathbb{R}$ be a measurable loss function. A learning problem $(\mathcal{H}, \mathcal{Z}, l)$ is
 811 called *Convex and ρ -Lipschitz* for $\rho > 0$ if:

- 812 - The hypothesis set \mathcal{H} is convex.
- 813 - For all $z \in \mathcal{Z}$, the (partially applied) loss function $l(\cdot; z)$ is convex and ρ -Lipschitz.

814 The original definition in [33] is about bounded hypothesis sets, but in our analysis we will use
 815 unbounded ones (as a regularization term in the loss function will effectively bound the solution
 816 space).

817 The learning algorithm will be Stochastic Gradient Descent (SGD) for ℓ_2 regularized learning
 818 problems (Algorithm 2 below). Note that the algorithm returns the average of the weights across T
 819 iterations, as it is typically the case in online learning.

Algorithm 2 Stochastic Gradient Descent (SGD) for minimizing $\mathbb{E}_{z \sim \mathcal{D}} [l(\mathbf{w}, z)] + \frac{\lambda}{2} \|\mathbf{w}\|^2$

Require: Integer $T > 0$

- 1: Initialize $\mathbf{w}^{(1)} = 0$
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: Sample $z \sim \mathcal{D}$
 - 4: Set $\mathbf{v}^{(t)} = \nabla_{\mathbf{w}^{(t)}} l(\mathbf{w}^{(t)}, z)$
 - 5: Set $\eta_t = \frac{1}{\lambda t}$
 - 6: Set $\mathbf{w}^{(t+\frac{1}{2})} = \mathbf{w}^{(t)} - \eta_t (\mathbf{v}^{(t)} + \lambda \mathbf{w}^{(t)})$
 - 7: **end for**
 - 8: Output $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)}$.
-

820 We study strongly convex learning objectives, as they permit tight bounds on the calibration of the
821 output of SGD later on. Analyzing their convex (only) counterparts would have yielded weaker
822 calibration guarantees – see, for instance, Chapter 4.7 in [27] and the analysis of early stopped SGD
823 in [48].

824 We now state a guarantee on the output of SGD after T iterations for convex and Lipschitz objectives
825 plus an additional ℓ_2 regularization term. The proof follows from Theorem 14.11 in [33] by applying it
826 to a learning objective and combining it with Markov’s inequality. We present it here for completeness.

827 **Theorem 3.** (Adaptation of Theorem 14.11 in [33]) Consider a Convex, ρ -Lipschitz learning problem
828 $(\mathcal{H} = \mathbb{R}^d, \mathcal{Z}, l)$ and a distribution \mathcal{D} over \mathcal{Z} . For every $\delta \in (0, 1)$, if we run the SGD method
829 (Algorithm 2) for minimizing $L_{\mathcal{D}, \lambda}(\mathbf{w}) = \mathbb{E}_{z \sim \mathcal{D}} [l(\mathbf{w}, z)] + \frac{\lambda}{2} \|\mathbf{w}\|^2$, $\lambda > 0$, for T iterations and
830 with learning rate $\eta_t = \frac{1}{\lambda t}$, then the output $\bar{\mathbf{w}}$ of SGD satisfies with probability at least $1 - \delta$ over
831 the sampling $z_1, \dots, z_T \sim \mathcal{D}$:

$$L_{\mathcal{D}, \lambda}(\bar{\mathbf{w}}) \leq \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w}) + \frac{2\rho^2}{\delta \lambda T} (1 + \ln T). \quad (13)$$

832 *Proof.* The objective $L_{\mathcal{D}, \lambda}(\mathbf{w}) = \mathbb{E}_{z \sim \mathcal{D}} [l(\mathbf{w}, z)] + \frac{\lambda}{2} \|\mathbf{w}\|^2$, $\lambda > 0$ is λ -strongly convex. Let $z \sim \mathcal{D}$,
833 then $\mathbf{v}^{(t)} = \nabla_{\mathbf{w}^{(t)}} l(\mathbf{w}^{(t)}, z)$ and, from the Lipschitz condition, it holds: $\|\mathbf{v}^{(t)}\| \leq \rho$. Let the update
834 direction be: $\mathbf{g}_t = \mathbf{v}^{(t)} + \lambda \mathbf{w}^{(t)}$. Then, we have for the weight vector $\mathbf{w}^{(t+1)}$:

$$\begin{aligned} \mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \eta_t (\mathbf{v}^{(t)} + \lambda \mathbf{w}^{(t)}) \\ &= \mathbf{w}^{(t)} \left(1 - \frac{1}{t}\right) - \frac{1}{\lambda t} \mathbf{v}^{(t)} \\ &= -\frac{1}{\lambda t} \sum_{i=1}^t \mathbf{v}^{(i)}. \end{aligned} \quad (14)$$

835 Hence, we have $\|\mathbf{w}^{(t)}\| \leq \frac{1}{\lambda(t-1)}(t-1)\rho = \frac{\rho}{\lambda}$, which implies that for the update vector it holds:
836 $\|\mathbf{g}_t\| \leq \rho + \lambda \frac{\rho}{\lambda} = 2\rho$. From Theorem 14.11 (SGD guarantee for strongly convex functions) in [33],
837 we get:

$$\mathbb{E}_{z_1, \dots, z_T \sim \mathcal{D}} [L_{\mathcal{D}, \lambda}(\bar{\mathbf{w}})] \leq \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w}) + \frac{(2\rho)^2}{2\lambda T} (1 + \ln T) = \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w}) + \frac{2\rho^2}{\lambda T} (1 + \ln T). \quad (15)$$

838 Finally, we apply Markov’s inequality on the non-negative random variable $L_{\mathcal{D}, \lambda}(\bar{\mathbf{w}}) -$
839 $\min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w})$. Let $0 < \delta < 1$, then by leveraging the above guarantee, we get:

$$\begin{aligned} \mathbb{P}_{z_1, \dots, z_T \sim \mathcal{D}} \left[L_{\mathcal{D}, \lambda}(\bar{\mathbf{w}}) - \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w}) \geq \tau \right] &\leq \frac{\mathbb{E}_{z_1, \dots, z_T \sim \mathcal{D}} [L_{\mathcal{D}, \lambda}(\bar{\mathbf{w}})] - \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}, \lambda}(\mathbf{w})}{\tau} \\ &\leq \frac{\frac{2\rho^2}{\lambda T} (1 + \ln T)}{\tau} < \delta, \end{aligned} \quad (16)$$

840 then with probability less than δ : $L_{\mathcal{D},\lambda}(\bar{\mathbf{w}}) - \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D},\lambda}(\mathbf{w}) \geq \tau > \frac{2\rho^2(1+\ln T)}{\lambda T \delta}$. In other words,
 841 with probability at least $1 - \delta$ over $z_1, \dots, z_T \sim \mathcal{D}$, it holds:

$$L_{\mathcal{D},\lambda}(\bar{\mathbf{w}}) \leq \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D},\lambda}(\mathbf{w}) + \frac{2\rho^2}{\delta \lambda T} (1 + \ln T). \quad (17)$$

842

□

843 *Remark 3.* We assumed that the loss function is differentiable, but the proof goes through for any
 844 continuous function using subgradients.

845 *Remark 4.* It is possible to tighten the δ dependency on the previous bound from $\Theta(1/\delta)$ to
 846 $\Theta(\ln(1/\delta))$ by using Azuma's instead of Markov's inequality.

847 F.1.1 Setup: Data distribution, Architecture and Learning Algorithms

848 **Data** As a reminder, we consider a parity mixture distribution. Let $\mathcal{X} = \{\pm 1\}^d$ be the input space
 849 of $d \geq 2$ bits and $\mathcal{Y} = \{\pm 1, \langle \text{EOS} \rangle\}^*$ be the output space of sequences, where $\langle \text{EOS} \rangle$ is a special
 850 symbol denoting the end of a string. Let $\mathcal{D}(p_{\text{cot}})$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, parameterized by
 851 $p_{\text{cot}} \in [0, 1]$, such that:

$$x_1, \dots, x_d \sim \text{Rad}(1/2)$$

$$(y_1, \dots, y_{d+1}) = Z(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle) + (1 - Z) \left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right), \quad (18)$$

852 where $Z \sim \text{Ber}(p_{\text{cot}})$.

853 **Model** We consider an architecture that consists of $d+1$ linear models. For the first output
 854 position, we consider the hypothesis class $\mathcal{H}_1 = \{\mathbf{x} \mapsto \langle \mathbf{w}_1, \mathbf{x} \rangle : \mathbf{x} \in \{\pm 1\}^d, \mathbf{w}_1 \in \mathbb{R}^d\}$.
 855 We break the second decision into two parts and consider two separate linear classes:
 856 $\mathcal{H}_{2a} = \{\mathbf{x} \mapsto \langle \mathbf{w}_{2a}, \phi_2(\mathbf{x}) \rangle + b_{2a} : \mathbf{x} \in \{\pm 1\}^{d+1}, \mathbf{w}_{2a} \in \mathbb{R}^{2d+1}, b_{2a} \in \mathbb{R}\}$ and $\mathcal{H}_2 =$
 857 $\{\mathbf{x} \mapsto \langle \mathbf{w}_2, \phi_2(\mathbf{x}) \rangle : \mathbf{x} \in \{\pm 1\}^{d+1}, \mathbf{w}_2 \in \mathbb{R}^{2d+1}\}$, where the first model decides between
 858 $\langle \text{EOS} \rangle$ and $\{\pm 1\}$ while the second model decides between -1 and $+1$, in case the first
 859 model did not predict $\langle \text{EOS} \rangle$. For the rest of the positions, we consider hypothesis classes
 860 $\mathcal{H}_l = \{\mathbf{x} \mapsto \langle \mathbf{w}_l, \phi_l(\mathbf{x}) \rangle : \mathbf{x} \in \{\pm 1\}^{2d+l-1}, \mathbf{w}_l \in \mathbb{R}^{2d+l-1}\}$. The feature embedding is defined as
 861 follows $\phi_l : \mathbf{x} \mapsto [x_1 \dots x_{d+l-1} \ x_{d+l-1} x_1 \dots x_{d+l-1} x_d]^T \in \{\pm 1\}^{2d+l-1}$ for $2 \leq l \leq d$.
 862 For position $d+1$, the output is a constant function of the input, as the output symbol is always
 863 $\langle \text{EOS} \rangle$. As learning is trivial in this case and, for not complicating the analysis any further, we
 864 assume access to this deterministic function $o : \{\pm 1\}^{2d} \mapsto \langle \text{EOS} \rangle$ for which it holds: $o(\mathbf{x}) = \langle \text{EOS} \rangle$
 865 for any $\mathbf{x} \in \{\pm 1\}^{2d}$.

866 Our final model is a function h that belongs to the linear autoregressive hypothesis class $\mathcal{H}_{\text{AR}}^{\text{Lin}}$, defined
 867 as:

$$\mathcal{H}_{\text{AR}}^{\text{Lin}} = \mathcal{H}_1 \times \mathcal{H}_{2a} \times \mathcal{H}_2 \dots \times \mathcal{H}_d. \quad (19)$$

868 Note that we can learn any sparse parity inside $\mathcal{H}_{\text{AR}}^{\text{Lin}}$, hence we argue that our definitions are not
 869 particularly tailored for the problem at hand. Given an $h \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ and the corresponding $d+1$ models,
 870 $\mathbf{w}_1, \boldsymbol{\theta}_{2a}, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_d$, we define a deterministic autoregressive process $\hat{h} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{2d+2} \times$

871 $\mathbb{R}^{2d+1} \times \dots \times \mathbb{R}^{3d-1} \mapsto \{-1, +1, \epsilon, \langle \text{EOS} \rangle\}^*$ as follows:

$$\begin{aligned}
\hat{h}_1(\mathbf{x}; \mathbf{w}_1) &= \text{sgn}(\langle \mathbf{w}_1, \mathbf{x} \rangle), \\
\hat{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) &= \text{dict} \left(\text{sgn} \left(\left\langle \mathbf{w}_{2a}, \phi_2 \left(\mathbf{x}, \hat{h}^{(1)}(\mathbf{x}; \mathbf{w}_1) \right) \right\rangle + b_{2a} \right) \right), \\
\boldsymbol{\theta}_{2a} &:= (\mathbf{w}_{2a}^T, b_{2a}), \\
\hat{h}_2(\mathbf{x}; \mathbf{w}_2) &= \begin{cases} \epsilon, & \text{if } \hat{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) = \langle \text{EOS} \rangle, \\ \text{sgn} \left(\left\langle \mathbf{w}_2, \phi_2 \left(\mathbf{x}, \hat{h}^{(1)}(\mathbf{x}; \mathbf{w}_1) \right) \right\rangle \right), & \text{o.w.,} \end{cases} \\
\hat{h}_3(\mathbf{x}; \mathbf{w}_3) &= \begin{cases} \epsilon, & \text{if } \hat{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) = \langle \text{EOS} \rangle, \\ \text{sgn} \left(\left\langle \mathbf{w}_3, \phi_3 \left(\mathbf{x}, \hat{h}^{(1)}(\mathbf{x}; \mathbf{w}_1), \hat{h}^{(2)}(\mathbf{x}; \mathbf{w}_2) \right) \right\rangle \right), & \text{o.w.,} \end{cases} \\
&\vdots \\
\hat{h}_d(\mathbf{x}; \mathbf{w}_d) &= \begin{cases} \epsilon, & \text{if } \hat{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) = \langle \text{EOS} \rangle, \\ \text{sgn} \left(\left\langle \mathbf{w}_d, \phi_d \left(\mathbf{x}, \hat{h}^{(1)}(\mathbf{x}; \mathbf{w}_1), \dots, \hat{h}^{(d-1)}(\mathbf{x}; \mathbf{w}_{d-1}) \right) \right\rangle \right), & \text{o.w.,} \end{cases} \\
\hat{h}_{d+1}(\mathbf{x}) &= \begin{cases} \epsilon, & \text{if } \hat{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) = \langle \text{EOS} \rangle, \\ o(\mathbf{x}), & \text{o.w.,} \end{cases} \\
\hat{h}(\mathbf{x}; \{\mathbf{w}_1, \boldsymbol{\theta}_{2a}, \mathbf{w}_2, \dots, \mathbf{w}_d\}) &= \left(\hat{h}^{(1)}(\mathbf{x}; \mathbf{w}_1), \dots, \hat{h}^{(d+1)}(\mathbf{x}) \right),
\end{aligned} \tag{20}$$

872 where $\text{dict} : \{\pm 1\} \mapsto \{\epsilon, \langle \text{EOS} \rangle\}$ is a deterministic function (dictionary) that maps label -1 to
873 the empty string and label $+1$ to the $\langle \text{EOS} \rangle$ token. Likewise, we define a randomized sequence to
874 sequence model \tilde{h} as follows:

$$\begin{aligned}
\tilde{h}_1(\mathbf{x}; \mathbf{w}_1) &\sim \text{Rad} \left(\frac{1}{1 + e^{-\langle \mathbf{w}_1, \mathbf{x} \rangle}} \right), \\
\tilde{h}_{2a}(\mathbf{x}; \boldsymbol{\theta}_{2a}) &\sim \text{dict} \left(\text{Rad} \left(\frac{1}{1 + e^{-\langle \mathbf{w}_{2a}, \phi_2(\mathbf{x}, \tilde{h}_1(\mathbf{x}; \mathbf{w}_1)) \rangle - b_{2a}}} \right) \right), \\
\boldsymbol{\theta}_{2a} &:= (\mathbf{w}_{2a}^T, b_{2a}), \\
\tilde{h}_2(\mathbf{x}; \mathbf{w}_2) &\sim \begin{cases} \epsilon, & \text{if } \tilde{h}_{2a}(\mathbf{x}) = \langle \text{EOS} \rangle, \\ \text{Rad} \left(\frac{1}{1 + e^{-\langle \mathbf{w}_2, \phi_2(\mathbf{x}, \tilde{h}_1(\mathbf{x}; \mathbf{w}_1)) \rangle}} \right), & \text{o.w.,} \end{cases} \\
\tilde{h}_3(\mathbf{x}; \mathbf{w}_3) &\sim \begin{cases} \epsilon, & \text{if } \tilde{h}_{2a}(\mathbf{x}) = \langle \text{EOS} \rangle, \\ \text{Rad} \left(\frac{1}{1 + e^{-\langle \mathbf{w}_3, \phi_3(\mathbf{x}, \tilde{h}_1(\mathbf{x}; \mathbf{w}_1), \tilde{h}_2(\mathbf{x}; \mathbf{w}_2)) \rangle}} \right), & \text{o.w.,} \end{cases} \\
&\vdots \\
\tilde{h}_d(\mathbf{x}; \mathbf{w}_d) &\sim \begin{cases} \epsilon, & \text{if } \tilde{h}_{2a}(\mathbf{x}) = \langle \text{EOS} \rangle, \\ \text{Rad} \left(\frac{1}{1 + e^{-\langle \mathbf{w}_d, \phi_d(\mathbf{x}, \tilde{h}_1(\mathbf{x}; \mathbf{w}_1), \dots, \tilde{h}_{d-1}(\mathbf{x}; \mathbf{w}_{d-1})) \rangle}} \right), & \text{o.w.,} \end{cases} \\
\tilde{h}_{d+1}(\mathbf{x}) &= \begin{cases} \epsilon, & \text{if } \tilde{h}_{2a}(\mathbf{x}) = \langle \text{EOS} \rangle, \\ o(\mathbf{x}), & \text{o.w.,} \end{cases} \\
\tilde{h}(\mathbf{x}; \{\mathbf{w}_1, \boldsymbol{\theta}_{2a}, \mathbf{w}_2, \dots, \mathbf{w}_d\}) &= \left(\tilde{h}_1(\mathbf{x}; \mathbf{w}_1), \dots, \tilde{h}_{d+1}(\mathbf{x}) \right).
\end{aligned} \tag{21}$$

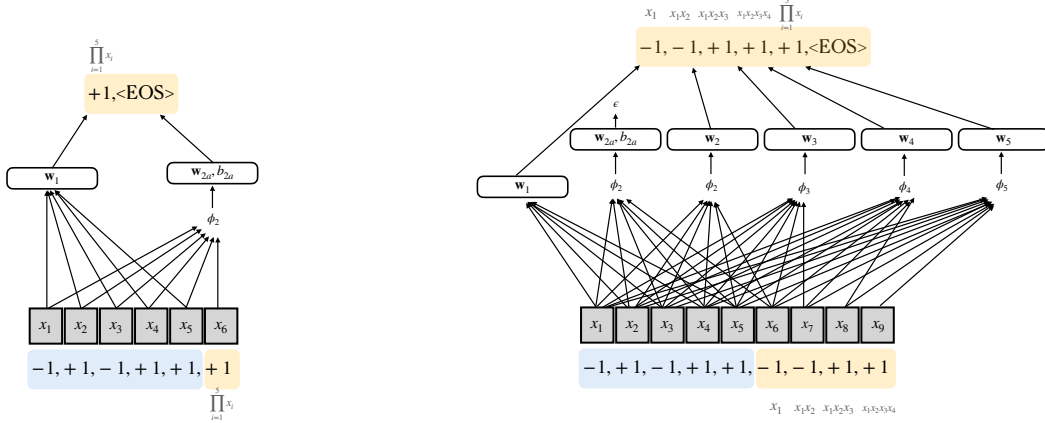


Figure 32: An illustration of next-token prediction training (for $d=5$) with the linear (time-inhomogeneous) autoregressive architecture in the case of a short (left) and long (right) training sequence.

We adopt the convention that $\alpha\epsilon\beta = \alpha\beta$, i.e., concatenation with the empty string character ϵ does not change a string. We define the probability measure induced by h as:

$$\pi_h(y | \mathbf{x}) = \begin{cases} \mathbb{P}[\tilde{h}_1(\mathbf{x}; \mathbf{w}_1) = y_1] \cdot \mathbb{P}[\tilde{h}_{2a}(\mathbf{x}; \theta_{2a}) = \langle \text{EOS} \rangle | \tilde{h}_1(\mathbf{x}; \mathbf{w}_1) = y_1], & \text{if } |y| = 2, \\ \mathbb{P}[\tilde{h}_1(\mathbf{x}; \mathbf{w}_1) = y_1] \cdot \dots \cdot \mathbb{P}[\tilde{h}_d(\mathbf{x}; \mathbf{w}_d) = y_d | \tilde{h}_1(\mathbf{x}; \mathbf{w}_1) = y_1, \dots, \\ \tilde{h}_{d-1}(\mathbf{x}; \mathbf{w}_{d-1}) = y_{d-1}], & \text{if } |y| > 2. \end{cases} \quad (22)$$

For any other $y \in \{\pm 1, \epsilon, \langle \text{EOS} \rangle\}^*$, it is $\pi_h(y | \mathbf{x}) = 0$.

The two autoregressive models, \hat{h} and \tilde{h} , correspond to *greedy decoding* and *sampling with temperature I* , respectively.

Pre-training loss function During pre-training, the loss function is the next token prediction objective, together with an ℓ_2 regularization term. Let $(\mathbf{x}, y) \sim \mathcal{D}(p_{\text{cot}})$. We define the loss functions corresponding to each linear model:

1. Position 1: $l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) = \ln(1 + e^{-y\langle \mathbf{w}_1, \mathbf{x} \rangle}) + \frac{\lambda_1}{2} \|\mathbf{w}_1\|^2$, $\lambda_1 > 0$.
2. Position 2a: $l^{(2a)}(\{\mathbf{w}_{2a}, b_{2a}\}, ((\mathbf{x}, y_1), \tilde{y}_2)) = \ln(1 + e^{-y(\langle \mathbf{w}_{2a}, \phi_2((\mathbf{x}, y_1)) \rangle + b_{2a})}) + \frac{\lambda_{2a}}{2} (\|\mathbf{w}_{2a}\|^2 + b_{2a}^2)$ with $\lambda_{2a} > 0$ and $\tilde{y}_2 = \begin{cases} +1, & y_2 = \langle \text{EOS} \rangle, \\ -1, & y_2 \in \{\pm 1\}. \end{cases}$
3. Positions 2 to d :

$$l^{(l)}(\mathbf{w}_l, ((\mathbf{x}, y_1, \dots, y_{l-1}), y_l)) = \ln(1 + e^{-y\langle \mathbf{w}_l, \phi_l((\mathbf{x}, y_1, \dots, y_{l-1})) \rangle}) + \frac{\lambda_l}{2} \|\mathbf{w}_l\|^2,$$

with $\lambda_l > 0$ for all $2 \leq l \leq d$.

We included the regularization term in the definition of the loss functions. Then, we seek to solve the following optimization problem:

$$\min_{\mathbf{w}_1, \mathbf{w}_{2a}, b, \mathbf{w}_2, \dots, \mathbf{w}_d} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}(p_{\text{cot}})} \left[\mathbb{1}\{|y| = 2\} \left(l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) + l^{(2a)}(\{\mathbf{w}_{2a}, b\}, ((\mathbf{x}, y_1), \tilde{y}_2)) \right) + \mathbb{1}\{|y| = d+1\} \left(l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) + \dots + l^{(d)}(\mathbf{w}_d, ((\mathbf{x}, y_1, \dots, y_{d-1}), y_d)) \right) \right]. \quad (\text{LIN-NTP})$$

889 Note that the regularization term can have a different coefficient for each parameter vector. Also,
 890 observe that Problem LIN-NTP corresponds to $d+1$ binary classification problems with respect to the
 891 logistic loss.

892 **Pre-training Algorithm** We minimize the previous objective with Stochastic Gradient Descent –
 893 see Algorithm 3.

Algorithm 3 Stochastic Gradient Descent (SGD) for solving Problem (LIN-NTP)

Require: Integers $T, T_1, T_{2a}, T_2, \dots, T_d > 0$, Real numbers $\lambda_1, \lambda_{2a}, \lambda_2, \dots, \lambda_d > 0$.

```

1: Initialize  $\mathbf{w}_1^{(1)}, \mathbf{w}_{2a}^{(1)}, \mathbf{w}_2^{(1)}, \dots, \mathbf{w}_d^{(1)} = \mathbf{0}, b_{2a} = 0$ 
2: Set  $t_{\text{long}} = 0$ 
3: for  $t = 1, 2, \dots, T$  do
4:   Sample  $(\mathbf{x}, y) \sim \mathcal{D}(p_{\text{cot}})$ 
5:   if  $t \leq T_1$  then
6:     Set  $\eta_t = \frac{1}{\lambda_1 t}$ 
7:     Set  $\mathbf{w}_1^{(t+1)} = \mathbf{w}_1^{(t)} - \eta_t \nabla_{\mathbf{w}_1^{(t)}} l^{(1)}(\mathbf{w}_1^{(t)}, (\mathbf{x}, y_1))$ 
8:   end if
9:   if  $t \leq T_{2a}$  then
10:    Set  $\tilde{y}_2 = \begin{cases} +1, & \text{if } y_2 = \text{<EOS>} \\ -1, & \text{if } y_2 \in \{\pm 1\} \end{cases}$ 
11:    Set  $\eta_t = \frac{1}{\lambda_{2a} t}$ 
12:    Set  $\mathbf{w}_{2a}^{(t+1)} = \mathbf{w}_{2a}^{(t)} - \eta_t \nabla_{\mathbf{w}_{2a}^{(t)}} l^{(2a)}(\{\mathbf{w}_{2a}^{(t)}, b_{2a}^{(t)}\}, ([\mathbf{x}, y_1], \tilde{y}_2))$ 
13:    Set  $b_{2a}^{(t+1)} = b_{2a}^{(t)} - \eta_t \frac{\partial}{\partial b_{2a}} l^{(2a)}(\{\mathbf{w}_{2a}^{(t)}, b_{2a}^{(t)}\}, ([\mathbf{x}, y_1], \tilde{y}_2))$ 
14:   end if
15:   if  $|y| > 2$  then
16:     Set  $t_{\text{long}} = t_{\text{long}} + 1$ 
17:     for  $l = 2, \dots, d$  do
18:       if  $t_{\text{long}} \leq T_l$  then
19:         Set  $\eta_{t_{\text{long}}} = \frac{1}{\lambda_l t_{\text{long}}}$ 
20:         Set  $\mathbf{w}_l^{(t_{\text{long}}+1)} = \mathbf{w}_l^{(t_{\text{long}})} - \eta_{t_{\text{long}}} \nabla_{\mathbf{w}_l^{(t_{\text{long}})}} l^{(l)}(\mathbf{w}_l^{(t_{\text{long}})}, ([\mathbf{x}, y_1, \dots, y_{l-1}], y_l))$ 
21:       end if
22:     end for
23:   end if
24: end for
25: Output  $\bar{\mathbf{w}}_1 = \frac{1}{T_1} \sum_{t=1}^{T_1} \mathbf{w}_1^{(t)}, \bar{\mathbf{w}}_{2a} = \frac{1}{T_{2a}} \sum_{t=1}^{T_{2a}} \mathbf{w}_{2a}^{(t)}, \bar{b}_{2a} = \frac{1}{T_{2a}} \sum_{t=1}^{T_{2a}} b_{2a}^{(t)}, \bar{\mathbf{w}}_2 = \frac{1}{T_2} \sum_{t=1}^{T_2} \mathbf{w}_2^{(t)}, \dots, \bar{\mathbf{w}}_d = \frac{1}{T_d} \sum_{t=1}^{T_d} \mathbf{w}_d^{(t)}$ .
```

894 **Post-training loss function** For post-training, we consider the STaR algorithm [49]. Recall that the
 895 STaR algorithm involves n reinforcement learning rounds, where each round involves optimization of
 896 a next-token prediction loss on model sampled responses (from the model of the previous round) that
 897 are correct according to some reward function. We use a reward $r_{\text{cot}} : \mathcal{X} \times \mathcal{Y}$ that assesses whether
 898 the whole sequence is valid:

$$r_{\text{cot}}(\mathbf{x}, y) = \mathbb{1} \left\{ y = \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \text{<EOS>} \right) \vee y = \left(\prod_{i=1}^d x_i, \text{<EOS>} \right) \right\}. \quad (23)$$

899 Namely, at the $k + 1$ round, we minimize the following objective:

$$\begin{aligned} \mathbb{E}_{\substack{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d} \\ y \sim \pi_{h^{(k)}}(\cdot | \mathbf{x})}} \left[\mathbb{1}\{|y| = 2\} \left(l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) + l^{(2a)}(\{\mathbf{w}_{2a}, b\}, ([\mathbf{x}, y_1], \tilde{y}_2)) \right) \right. \\ \left. + \mathbb{1}\{|y| = d + 1\} \left(l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) + l^{(2a)}(\{\mathbf{w}_{2a}, b\}, ([\mathbf{x}, y_1], \tilde{y}_2)) + \dots \right. \right. \\ \left. \left. + \dots + l^{(d)}(\mathbf{w}_d, ([\mathbf{x}, y_1, \dots, y_{d-1}], y_d)) \right) \right] \Big|_{r_{\text{cot}}(\mathbf{x}, y) = 1}, \end{aligned} \quad (\text{LIN-RL})$$

900 where $h^{(k)}$ is the model returned at the end of the k 'th round.

901 **Post-training Algorithm** The algorithm consists of n rounds, where at each one we minimize the
 902 previous objective with Stochastic Gradient Descent. At each round, we start optimization from a
 903 freshly initialized model at the origin.

904 F.2 Pre-training

905 We proceed by providing guarantees for each one of the models $\bar{\mathbf{w}}_1, \bar{\theta}_{2a}, \bar{\mathbf{w}}_2, \bar{\mathbf{w}}_3, \dots, \bar{\mathbf{w}}_d$ indepen-
 906 dently, and then state and prove our main pre-training theorem for the behavior of induced models
 907 \hat{h}, \tilde{h} , leveraging the per-position results. All results are stated for a mixture coefficient $p \in (0, 3/4)$.
 908 The upper bound needs to be greater than $1/2$ to obtain the post-training result later on, but the
 909 specific $3/4$ value is arbitrary.

910 F.2.1 Position 1

911 For the first position of the output, we consider a binary classification problem where $\mathcal{X}_1 =$
 912 $\{\pm 1\}^d, \mathcal{Y}_1 = \{\pm 1\}$ with a distribution $\mathcal{D}_1(p)$ over $\mathcal{X}_1 \times \mathcal{Y}_1$ such that: $x_1, \dots, x_d \sim \text{Rad}(1/2)$
 913 and $y = Zx_1 + (1 - Z) \prod_{i=1}^d x_i$ where $Z \sim \text{Ber}(p)$, $0 < p < 3/4$. We consider the hypothesis
 914 class $\mathcal{H}_1 = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathbb{R}^d\}$ and the logistic loss plus an additional ℓ_2 regularization term:
 915 $l^{(1)}(\mathbf{w}, (\mathbf{x}, y)) = \ln(1 + e^{-y\langle \mathbf{w}, \mathbf{x} \rangle}) + \frac{\lambda_1}{2} \|\mathbf{w}\|^2$, $\lambda_1 > 0$.

916 We prove the following guarantees on the hypothesis returned by SGD.

917 **Proposition 1.** Consider running SGD (Algorithm 2) for minimizing $L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w}) =$
 918 $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} [\ln(1 + e^{-y\langle \mathbf{w}, \mathbf{x} \rangle})] + \frac{\lambda_1}{2} \|\mathbf{w}\|^2$ with $\lambda_1 > 0$. Then, after T_1 iterations and for any
 919 $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_1} \sim \mathcal{D}_1(p)$, it holds for all
 920 $\mathbf{x} \in \{\pm 1\}^d$:

$$\begin{aligned} \left| \left\langle \bar{\mathbf{w}}_1 - \ln\left(\frac{1+p}{1-p}\right) \mathbf{e}_1, \mathbf{x} \right\rangle \right| &\leq \frac{2d}{\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{4 \ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1, \\ \left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle}} - \frac{1}{1 + \left(\frac{1-p}{1+p}\right)^{x_1}} \right| &< \frac{d}{2\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{\ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1, \end{aligned} \quad (24)$$

921 where $\bar{\mathbf{w}}_1$ is the output of SGD and $\mathbf{e}_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^d$.

922 *Proof.* We first show that learning \mathcal{H}_1 with respect to (unregularized loss) $\ln(1 + e^{-y\langle \mathbf{w}, \mathbf{x} \rangle})$
 923 corresponds to a Convex and Lipschitz learning problem. The loss is convex with respect to its first
 924 argument. For the Lipschitz constant, we have for all $\mathbf{x} \in \{\pm 1\}^d, y \in \{\pm 1\}$ and $\mathbf{w} \in \mathbb{R}^d$:

$$\left\| \nabla_{\mathbf{w}} \ln(1 + e^{-y\langle \mathbf{w}, \mathbf{x} \rangle}) \right\| = \left\| -\frac{y\mathbf{x}}{1 + e^{y\langle \mathbf{w}, \mathbf{x} \rangle}} \right\| \leq \sqrt{d}. \quad (25)$$

925 Therefore, applying Theorem 3, we have that SGD after T_1 iterations returns a hypothesis $\bar{\mathbf{w}}_1$ such
 926 that for any $\delta \in (0, 1)$ with probability at least $1 - \delta$ it holds:

$$L_{\mathcal{D}_1(p), \lambda_1}(\bar{\mathbf{w}}_1) \leq L_{\mathcal{D}_1(p), \lambda_1}(\hat{\mathbf{w}}) + \frac{2d}{\delta \lambda_1 T_1} (1 + \ln T_1), \quad (26)$$

927 where $\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathcal{H}_1} L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w})$. From the strong convexity of $L_{\mathcal{D}_1(p), \lambda_1}$, this implies:

$$\|\bar{\mathbf{w}}_1 - \hat{\mathbf{w}}\|^2 \leq \frac{2}{\lambda_1} (L_{\mathcal{D}_1(p), \lambda_1}(\bar{\mathbf{w}}_1) - L_{\mathcal{D}_1(p), \lambda_1}(\hat{\mathbf{w}})) \leq \frac{4d}{\lambda_1^2 \delta T_1} (1 + \ln T_1). \quad (27)$$

928 The previous bound on the parameter space can be translated to a guarantee on the estimated
 929 probability of the output being equal to 1. Recall that for a hypothesis \mathbf{w} , this probability is defined
 930 as the output of the hypothesis passed through the sigmoid function $\sigma(u) = \frac{1}{1+e^{-u}}$:

$$p_1(\mathbf{w} \mid \mathbf{x}) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle) = \frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}. \quad (28)$$

931 We calculate the Lipschitz constant of $p_1(\mathbf{w} \mid \mathbf{x})$. For any $\mathbf{x} \in \{\pm 1\}^d$, $\mathbf{w} \in \mathbb{R}^d$, we have:

$$\|\nabla_{\mathbf{w}} p_1(\mathbf{w} \mid \mathbf{x})\| = \left\| \frac{\mathbf{x} e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}{(1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle})^2} \right\| \leq \frac{\sqrt{d}}{4}. \quad (29)$$

932 Therefore, combining eqs. 27, 29, we have:

$$|p_1(\bar{\mathbf{w}}_1 \mid \mathbf{x}) - p_1(\hat{\mathbf{w}} \mid \mathbf{x})| \leq \frac{\sqrt{d}}{4} \|\bar{\mathbf{w}}_1 - \hat{\mathbf{w}}\| \leq \frac{d}{2\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}}. \quad (30)$$

933 It remains to estimate the value of $p_1(\hat{\mathbf{w}} \mid \mathbf{x})$. In order to find the minimizer $\hat{\mathbf{w}} =$
 934 $\arg \min_{\mathbf{w} \in \mathcal{H}_1} L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w})$, we set the gradient of $L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w})$ to zero:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} \left[\frac{-y\mathbf{x}}{1 + e^{y\langle \hat{\mathbf{w}}, \mathbf{x} \rangle}} \right] + \lambda_1 \hat{\mathbf{w}} = 0. \quad (31)$$

935 The objective $L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w})$ is strongly convex, so it admits a unique solution. We observe that
 936 this solution is of the form $\hat{\mathbf{w}} = \alpha e_1$, $\alpha \in \mathbb{R}$, where $e_1 = [1, 0, \dots, 0]^T$. Indeed, the optimality
 937 conditions become:

$$\begin{cases} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} \left[\frac{-yx_1}{1 + e^{\alpha y x_1}} \right] + \lambda_1 \alpha = 0, \\ \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} \left[\frac{-yx_i}{1 + e^{\alpha y x_1}} \right] = 0, i = 2, \dots, d. \end{cases} \quad (32)$$

938 The last $d-1$ equations are satisfied as $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} [yx_i] = 0$ for all $i = 2, \dots, d$. The first equation
 939 simplifies to:

$$g(\alpha) := \frac{1}{1 + e^{-\alpha}} + \lambda_1 \alpha - \frac{p+1}{2} = 0. \quad (33)$$

940 This equation has, indeed, a unique root as g is continuous, $g'(u) = \frac{e^{-u}}{(1+e^{-u})^2} + \lambda_1 > 0$ for all
 941 $\lambda_1 > 0$, $u \in \mathbb{R}$ and $\lim_{u \rightarrow -\infty} g(u) = -\infty$, $\lim_{u \rightarrow +\infty} g(u) = +\infty$. Furthermore, $g(0) = -\frac{p}{2} < 0$,
 942 hence $\alpha > 0$. This proves that $\hat{\mathbf{w}} = \alpha e_1$, where $\alpha > 0$ is such that $g(\alpha) = 0$. Furthermore,
 943 let $\alpha_0 = \ln\left(\frac{1+p}{1-p}\right)$ be the weight of the unregularized solution $\hat{\mathbf{w}}_0$; that is, α_0 is the solution of
 944 $g(\alpha_0) = 0$ for $\lambda_1 = 0$. We have $g(\alpha) - g(\alpha_0) = -\lambda_1 \alpha < 0$ which implies that $\alpha < \alpha_0$. From the
 945 mean value theorem there exists $\xi \in (\alpha, \alpha_0)$ such that:

$$\sigma'(\xi) = \frac{\sigma(\alpha_0) - \sigma(\alpha)}{\alpha_0 - \alpha} = \frac{\lambda_1 \alpha}{\alpha_0 - \alpha}. \quad (34)$$

946 But, observe that for any $u \in [0, \alpha_0]$, the derivative of the sigmoid is bounded as follows: $\frac{1-p^2}{4} \leq$
 947 $\sigma'(u) \leq \frac{1}{4}$. Therefore, for any $\lambda_1 > 0$, it holds:

$$|\alpha - \alpha_0| \leq \frac{4\alpha}{1-p^2} \lambda_1 < \frac{4\alpha_0}{1-p^2} \lambda_1 = \frac{4 \ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1. \quad (35)$$

948 Given the above, we can bound the calibration of $\hat{\mathbf{w}}$. For any $\lambda_1 > 0$ and $\mathbf{x} \in \{\pm 1\}^d$ we have:

$$\begin{aligned} \left| \frac{1}{1 + e^{-\alpha x_1}} - \frac{1}{1 + e^{-\alpha_0 x_1}} \right| &\leq \frac{1}{4} |\alpha - \alpha_0| \\ &< \frac{\ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1. \end{aligned} \quad (36)$$

949 Combining the above with (30), we finally obtain that for any $\lambda_1 > 0$ and $\mathbf{x} \in \{\pm 1\}^d$, it holds:

$$\begin{aligned} \left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle}} - \frac{1}{1 + \left(\frac{1-p}{1+p}\right)^{x_1}} \right| &= |p_1(\bar{\mathbf{w}} \mid \mathbf{x}) - p_1(\hat{\mathbf{w}}_0 \mid \mathbf{x})| \\ &\leq |p_1(\bar{\mathbf{w}} \mid \mathbf{x}) - p_1(\hat{\mathbf{w}} \mid \mathbf{x})| + |p_1(\hat{\mathbf{w}} \mid \mathbf{x}) - p_1(\hat{\mathbf{w}}_0 \mid \mathbf{x})| \\ &< \frac{d}{2\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{\ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1. \end{aligned} \quad (37)$$

950 Furthermore, for the optimality of $\bar{\mathbf{w}}_1$ in parameter space, we have:

$$\begin{aligned} |\langle \bar{\mathbf{w}}_1 - \alpha_0 \mathbf{e}_1, \mathbf{x} \rangle| &= |\langle \bar{\mathbf{w}}_1 - \alpha \mathbf{e}_1, \mathbf{x} \rangle - \langle \alpha_0 \mathbf{e}_1 - \alpha \mathbf{e}_1, \mathbf{x} \rangle| \\ &\leq \sqrt{d} \|\bar{\mathbf{w}}_1 - \hat{\mathbf{w}}\| + |\alpha - \alpha_0|. \end{aligned} \quad (38)$$

951 From eqs. 27 and 35, we obtain:

$$|\langle \bar{\mathbf{w}}_1 - \alpha_0 \mathbf{e}_1, \mathbf{x} \rangle| \leq \frac{2d}{\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{4 \ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1. \quad (39)$$

952 \square

953 The guarantees of (24) quantify the calibration of the model returned by SGD and its proximity to the
954 best-in-class model in parameter space. We obtain the following explicit corollary on the calibration
955 and “hard” prediction of the model.

956 **Corollary 1.** *Consider running SGD (Algorithm 2) for minimizing $L_{\mathcal{D}_1(p), \lambda_1}(\mathbf{w}) =$
957 $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_1(p)} [\ln(1 + e^{-y \langle \mathbf{w}, \mathbf{x} \rangle})] + \frac{\lambda_1}{2} \|\mathbf{w}\|^2$. Then, for any $\varepsilon > 0$, if $\lambda_1 = \tilde{\Theta}\left(\frac{d^{1/2}}{\delta^{1/4} T_1^{1/4}}\right)$, and after
958 $T_1 = \tilde{O}\left(\frac{d^2}{\delta \varepsilon^4}\right)$ iterations, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_1} \sim \mathcal{D}_1(p)$, it
959 holds:*

$$\left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle}} - \frac{1}{1 + \left(\frac{1-p}{1+p}\right)^{x_1}} \right| < \varepsilon, \quad (40)$$

960 for all $\mathbf{x} \in \{\pm 1\}^d$. If, additionally, $\varepsilon < \ln\left(\frac{1+p}{1-p}\right)$, then with probability $1 - \delta$:

$$\text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle) = x_1, \quad (41)$$

961 for all $\mathbf{x} \in \{\pm 1\}^d$.

962 *Proof.* Recall that from eq. (24), with probability $1 - \delta$ we have:

$$\begin{aligned} \left| \left\langle \bar{\mathbf{w}}_1 - \ln\left(\frac{1+p}{1-p}\right) \mathbf{e}_1, \mathbf{x} \right\rangle \right| &\leq \frac{2d}{\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{4 \ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1, \\ \left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle}} - \frac{1}{1 + \left(\frac{1-p}{1+p}\right)^{x_1}} \right| &< \frac{d}{2\lambda_1} \sqrt{\frac{1 + \ln T_1}{\delta T_1}} + \frac{\ln\left(\frac{1+p}{1-p}\right)}{1-p^2} \lambda_1. \end{aligned} \quad (42)$$

963 To instantiate the bounds, first observe that for any $0 < p < 3/4$, it holds: $\frac{\ln\left(\frac{1+p}{1-p}\right)}{1-p^2} < \frac{16 \ln 7}{7}$.

964 Furthermore, the first bound is always larger than the second one. We set $\lambda_1 = \sqrt{\frac{d\left(\frac{1+\ln T_1}{T_1}\right)^{1/2}}{\frac{32 \ln 7}{7}}}$ to
965 bound the two terms and solve the following inequality for T_1 :

$$\begin{aligned} d^{1/2} \left(\frac{1 + \ln T_1}{\delta T_1} \right)^{1/4} \left(\frac{512 \ln 7}{7} \right)^{1/2} &< \varepsilon \\ \frac{T_1}{1 + \ln T_1} &> \frac{512^2}{49} (\ln 7)^2 \frac{d^2}{\delta \varepsilon^4}. \end{aligned} \quad (43)$$

966 This proves the first part of the claim. For the decision term $\text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle)$, we have:

$$\begin{aligned} \text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle) &= \text{sgn} \left(\left\langle \bar{\mathbf{w}}_1 - \ln \left(\frac{1+p}{1-p} \right) \mathbf{e}_1, \mathbf{x} \right\rangle + \left\langle \ln \left(\frac{1+p}{1-p} \right) \mathbf{e}_1, \mathbf{x} \right\rangle \right) \\ &= \text{sgn} \left(\left\langle \bar{\mathbf{w}}_1 - \ln \left(\frac{1+p}{1-p} \right) \mathbf{e}_1, \mathbf{x} \right\rangle + x_1 \ln \left(\frac{1+p}{1-p} \right) \right). \end{aligned} \quad (44)$$

967 If $\left| x_1 \ln \left(\frac{1+p}{1-p} \right) \right| > \left| \left\langle \bar{\mathbf{w}}_1 - \ln \left(\frac{1+p}{1-p} \right) \mathbf{e}_1, \mathbf{x} \right\rangle \right|$, then the following holds: $\text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle) =$
 968 $\text{sgn} \left(x_1 \ln \left(\frac{1+p}{1-p} \right) \right) = x_1$. For this to happen, we additionally require having $\varepsilon < \ln \left(\frac{1+p}{1-p} \right)$. \square

969 F.2.2 Position 2

970 **Position 2a** We consider a binary classification problem where $\mathcal{X}_{2a} = \{\pm 1\}^{d+1}$, $\mathcal{Y}_{2a} = \{\pm 1\}$
 971 with a distribution $\mathcal{D}_{2a}(p)$ over $\mathcal{X}_{2a} \times \mathcal{Y}_{2a}$ such that: $x_1, \dots, x_d \sim \text{Rad}(1/2)$ and $(x_{d+1}, y) =$
 972 $Z(x_1, -1) + (1-Z) \left(\prod_{i=1}^d x_i, +1 \right)$ where $Z \sim \text{Ber}(p)$, $0 < p < 3/4$. The -1 label should be inter-
 973 preted as the empty string ϵ , while the $+1$ output corresponds to the $\langle \text{EOS} \rangle$ symbol. We consider a non-
 974 homogeneous linear hypothesis class $\mathcal{H}_{2a} = \{ \mathbf{x} \mapsto \langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b : \mathbf{w} \in \mathbb{R}^{2d+1}, b \in \mathbb{R} \}$, where
 975 the feature map $\phi_2 : \mathbf{x} \mapsto [x_1 \dots x_{d+1} \ x_1 x_{d+1} \ x_2 x_{d+1} \dots x_d x_{d+1}]^T \in \{\pm 1\}^{2d+1}$ aug-
 976 ments the input with the second degree monomials that involve the inputs bits and the bit from the
 977 previous position. In this way, the linear model is capable of approximating the target via the $x_1 x_{d+1}$
 978 feature which can help predict whether the generation is on the “long” or “short” path. Note that the
 979 dimension of the feature map $-O(d)$ – is still polynomial in the input dimension. The bias term is
 980 crucial for representing the best-in-class conditional probabilities. We consider learning with the
 981 ℓ_2 -regularized logistic loss:

$$l^{(2a)}(\boldsymbol{\theta} = \{\mathbf{w}, b\}, (\mathbf{x}, y)) = \ln \left(1 + e^{-y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)} \right) + \frac{\lambda_{2a}}{2} (\|\mathbf{w}\|^2 + b^2), \lambda_{2a} > 0. \quad (45)$$

982 **Proposition 2.** Consider running SGD (Algorithm 2) for minimizing $L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\boldsymbol{\theta} = \{\mathbf{w}, b\}) =$
 983 $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_{2a}(p)} [\ln(1 + e^{-y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)})] + \frac{\lambda_{2a}}{2} (\|\mathbf{w}\|^2 + b^2)$, $\lambda_{2a} > 0$. Then, after T_{2a} iterations
 984 and for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_{2a}}$, it holds for all
 985 $\mathbf{x} \in \{\pm 1\}^{d+1}$ with $x_1 x_{d+1} = -1$:

$$\frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle + \bar{b}_{2a}}} < \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \sqrt{\frac{\lambda_{2a}}{1-p}}, \quad (46)$$

986 while for $\mathbf{x} \in \{\pm 1\}^{d+1}$ such that $x_1 x_{d+1} = +1$, it holds:

$$\begin{aligned} \left| \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle - \bar{b}_{2a}}} - \frac{1-p}{1+p} \right| &< \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p)}{8p(1-p)} \left| \ln \left(\frac{2p}{1-p} \right) \right|, \\ \left| \langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 \right| &< \frac{8(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p)}{2p(1-p)} \left| \ln \left(\frac{2p}{1-p} \right) \right|, \end{aligned} \quad (47)$$

987 where $\bar{\mathbf{w}}_{2a}, \bar{b}_{2a}$ is the output of SGD and $\hat{a}_0, \hat{b}_0 \in \mathbb{R}$ such that: $\hat{a}_0 + \hat{b}_0 = \ln \left(\frac{1-p}{2p} \right)$.

988 *Proof.* We first show that learning \mathcal{H}_{2a} with respect to the unregularized loss
 989 $\ln(1 + e^{-y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)})$ corresponds to a Convex and Lipschitz learning problem. We
 990 will abuse notation and write $\boldsymbol{\theta} \in \mathcal{H}_{2a}$ for $\boldsymbol{\theta} = (\mathbf{w}, b)$, where $(\mathbf{w}, b) \in \mathcal{H}_{2a}$. The loss is convex
 991 with respect to $\boldsymbol{\theta}$. For the Lipschitz constant, we bound the gradient with respect to $\boldsymbol{\theta}$. For all
 992 $\mathbf{x} \in \{\pm 1\}^{d+1}$, $y \in \{\pm 1\}$, we have:

$$\nabla_{\mathbf{w}} l^{(2a)}(\boldsymbol{\theta}, (\mathbf{x}, y)) = -\frac{y \phi_2(\mathbf{x})}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}}, \quad (48)$$

993 and for the bias term:

$$\frac{\partial}{\partial b} l^{(2a)}(\boldsymbol{\theta}, (\mathbf{x}, y)) = -\frac{y}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}}. \quad (49)$$

As a result, we have for the Lipschitz constant:

$$\left\| \nabla_{\theta} l^{(2a)}(\theta, (\mathbf{x}, y)) \right\|^2 = \left\| -\frac{y\phi_2(\mathbf{x})}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}} \right\|^2 + \left(-\frac{y}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}} \right)^2 \leq (2d+1) + 1 \quad (50)$$

which implies that the function is $\sqrt{2d+2}$ -Lipschitz with respect to θ . Therefore, applying Theorem 3, we have that SGD after T_{2a} iterations returns a hypothesis $\bar{\theta}_{2a}$ such that for any $\delta \in (0, 1)$ with probability at least $1 - \delta$ it holds:

$$L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\bar{\theta}_{2a}) \leq L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\hat{\theta}) + \frac{4(d+1)}{\lambda_{2a}\delta T_{2a}} (1 + \ln T_{2a}), \quad (51)$$

where $\hat{\theta} = \arg \min_{\theta \in \mathcal{H}_{2a}} L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\theta)$. From the strong convexity of $L_{\mathcal{D}_{2a}(p), \lambda_{2a}}$, this implies:

$$\left\| \bar{\theta}_{2a} - \hat{\theta} \right\|^2 \leq \frac{2}{\lambda_{2a}} \left(L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\bar{\theta}_{2a}) - L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\hat{\theta}) \right) \leq \frac{8(d+1)}{\lambda_{2a}^2 \delta T_{2a}} (1 + \ln T_{2a}). \quad (52)$$

We characterize now the loss minimizer $\hat{\theta} = \{\hat{\mathbf{w}}, \hat{b}\}$ by setting the gradient of $L_{\mathcal{D}_{2a}(p), \lambda_{2a}}$ to zero:

$$\begin{cases} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_{2a}(p)} \left[\frac{-y\phi_2(\mathbf{x})}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}} \right] + \lambda_{2a} \hat{\mathbf{w}} = 0, \\ \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_{2a}(p)} \left[-\frac{y}{1 + e^{y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)}} \right] + \lambda_{2a} \hat{b} = 0. \end{cases} \quad (53)$$

The objective $L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\theta)$ is strongly convex, so it admits a unique solution. We observe that this solution is of the form $\hat{\mathbf{w}} = \hat{a}e_{d+2}$, $\hat{a} \in \mathbb{R}$, where $e_{d+2} = [0, \dots, 0, 1, 0, \dots, 0]^T$ and $\hat{b} \in \mathbb{R}$. Indeed, the optimality conditions yield:

$$\begin{cases} \frac{1}{1 + e^{-(\hat{a} - \hat{b})}} + \frac{\lambda_{2a}}{1-p} (\hat{a} - \hat{b}) = 0, \\ \frac{2p}{1 + e^{-(\hat{a} + \hat{b})}} + \frac{p-1}{1 + e^{\hat{a} + \hat{b}}} + \lambda_{2a} (\hat{a} + \hat{b}) = 0. \end{cases} \quad (54)$$

The functions $g_1(u) = \frac{1}{1 + e^{-u}} + \lambda u$, $\lambda > 0$ and $g_2(u) = \frac{2p}{1 + e^{-u}} + \frac{p-1}{1 + e^u} + \lambda u$, $\lambda > 0$ are continuous, monotonically increasing and $\lim_{u \rightarrow -\infty} g_1(u) = \lim_{u \rightarrow -\infty} g_2(u) = -\infty$, $\lim_{u \rightarrow +\infty} g_1(u) = \lim_{u \rightarrow +\infty} g_2(u) = +\infty$, hence they have unique roots. This proves that, indeed, $\hat{\mathbf{w}} = \hat{a}e_{d+2}$, where \hat{a}, \hat{b} are such that $g_1(\hat{a} - \hat{b}) = 0$ and $g_2(\hat{a} + \hat{b}) = 0$. We will now bound the error in the predicted probabilities, depending on what region of the distribution the input lies. We consider the following cases:

• If $x_1 x_{d+1} = -1$, then we have:

$$\begin{aligned} p_{-1} \left(\bar{\theta}_{2a} \middle| x_1 x_{d+1} = -1 \right) &= \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a}}} \\ &= \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a}}} - \frac{1}{1 + e^{\langle \hat{\mathbf{w}}, \phi_2(\mathbf{x}) \rangle + \hat{b}}} + \frac{1}{1 + e^{\langle \hat{\mathbf{w}}, \phi_2(\mathbf{x}) \rangle + \hat{b}}} \\ &< \frac{\sqrt{2(d+1)}}{4} \left\| \bar{\theta}_{2a} - \hat{\theta} \right\| + \frac{1}{1 + e^{\langle \hat{\mathbf{w}}, \phi_2(\mathbf{x}) \rangle + \hat{b}}} \\ &\leq \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{1}{1 + e^{\langle \hat{\mathbf{w}}, \phi_2(\mathbf{x}) \rangle + \hat{b}}}, \end{aligned} \quad (55)$$

where we used (52) and the Lipschitzness of the logistic function. For the bias error term, we have:

$$\begin{aligned} \frac{1}{1 + e^{\langle \hat{\mathbf{w}}, \phi_2(\mathbf{x}) \rangle + \hat{b}}} &= \frac{1}{1 + e^{\hat{a} x_1 x_{d+1} + \hat{b}}} \\ &= \frac{1}{1 + e^{-(\hat{a} - \hat{b})}} \\ &= -\frac{\lambda_{2a}}{1-p} (\hat{a} - \hat{b}) \quad (\text{from (54)}) \\ &< \sqrt{\frac{\lambda_{2a}}{1-p}}, \end{aligned} \quad (56)$$

1012 as the solution $\hat{a} - \hat{b}$ of the $g_1(u) = 0$ equation lies in $\left(-\sqrt{\frac{1-p}{\lambda_{2a}}}, 0\right)$.

1013 • If $x_1 x_{d+1} = 1$, then denote by $\hat{a}_0 + \hat{b}_0$ the solution of the unregularized problem $g_2(\hat{a}_0 +$
 1014 $\hat{b}_0) = 0$ for $\lambda = 0$ or, equivalently from (54), $\hat{a}_0 + \hat{b}_0 = \ln\left(\frac{1-p}{2p}\right)$. We bound the distance
 1015 between the model-induced probabilities and the best in-class probabilities:

$$\begin{aligned} & \left| p_1\left(\bar{\theta}_{2a} \middle| x_1 x_{d+1} = +1\right) - p_1\left(\hat{\theta}_0 \middle| x_1 x_{d+1} = +1\right) \right| \\ & \leq \left| p_1\left(\bar{\theta}_{2a} \middle| x_1 x_{d+1} = +1\right) - p_1\left(\hat{\theta} \middle| x_1 x_{d+1} = +1\right) \right| \\ & \quad + \left| p_1\left(\hat{\theta} \middle| x_1 x_{d+1} = +1\right) - p_1\left(\hat{\theta}_0 \middle| x_1 x_{d+1} = +1\right) \right| \quad (57) \\ & = \frac{\sqrt{2(d+1)}}{4} \left\| \bar{\theta}_{2a} - \hat{\theta} \right\| + \frac{1}{4} \left| (\hat{a} + \hat{b}) - (\hat{a}_0 + \hat{b}_0) \right|. \end{aligned}$$

1016 It remains to show that the distance $\left| (\hat{a} + \hat{b}) - (\hat{a}_0 + \hat{b}_0) \right|$ is bounded. Let us denote the
 1017 unregularized part of g_2 as \bar{g} , that is $\bar{g}(u) = \frac{2p}{1+e^{-u}} + \frac{p-1}{1+e^u}$. First, observe that $g_2(0) = \frac{3p-1}{2}$,
 1018 which means that the sign of $\hat{a} + \hat{b}$ depends on the value of p . It will be easier to treat these
 1019 three subcases separately:

1020 – If $p < 1/3$, then $g_2(0) < 0$ and $g_2(\hat{a}_0 + \hat{b}_0) = \lambda(\hat{a}_0 + \hat{b}_0) > 0$, so it holds
 1021 $0 < \hat{a} + \hat{b} < \hat{a}_0 + \hat{b}_0$. From the mean value theorem, there exists $\xi \in (\hat{a} + \hat{b}, \hat{a}_0 + \hat{b}_0)$
 1022 such that:

$$\bar{g}'(\xi) = \frac{\lambda_{2a}(\hat{a} + \hat{b})}{(\hat{a}_0 + \hat{b}_0) - (\hat{a} + \hat{b})}. \quad (58)$$

1023 But, $\bar{g}'(u) = \frac{(p+1)e^u}{(1+e^u)^2}$ and, in particular, $\bar{g}'(\hat{a}_0 + \hat{b}_0) = \bar{g}'\left(\ln\left(\frac{1-p}{2p}\right)\right) = \frac{2p(1-p)}{1+p}$.

1024 This implies that for any $u \in \left(0, \ln\left(\frac{1-p}{2p}\right)\right)$, it holds: $\frac{2p(1-p)}{1+p} \leq \bar{g}'(u) \leq \frac{1+p}{4}$. Thus,

$$\begin{aligned} (\hat{a}_0 + \hat{b}_0) - (\hat{a} + \hat{b}) &= \frac{\lambda_{2a}(\hat{a} + \hat{b})}{\bar{g}'(\xi)} \\ &< \frac{\lambda_{2a}(1+p)(\hat{a}_0 + \hat{b}_0)}{2p(1-p)} \quad (59) \\ &= \frac{\lambda_{2a}(1+p)\ln\left(\frac{1-p}{2p}\right)}{2p(1-p)}. \end{aligned}$$

1025 – If $p = 1/3$, then $\hat{a} + \hat{b} = 0$ as $g_2(0) = 0$ and, also, $\hat{a}_0 + \hat{b}_0 = \ln\left(\frac{1-p}{2p}\right) = 0$. Hence,
 1026 the bias error term is 0.

1027 – If $p > 1/3$, then $g_2(0) > 0$, $g_2(\hat{a}_0 + \hat{b}_0) = \lambda_{2a}(\hat{a}_0 + \hat{b}_0) < 0$, so $\hat{a}_0 + \hat{b}_0 < \hat{a} + \hat{b} < 0$.

1028 From the mean value theorem, there exists $\xi \in (\hat{a}_0 + \hat{b}_0, \hat{a} + \hat{b})$ such that:

$$\bar{g}'(\xi) = -\frac{\lambda_{2a}(\hat{a} + \hat{b})}{(\hat{a} + \hat{b}) - (\hat{a}_0 + \hat{b}_0)} < -\frac{\lambda_{2a}(\hat{a}_0 + \hat{b}_0)}{(\hat{a} + \hat{b}) - (\hat{a}_0 + \hat{b}_0)}, \quad (60)$$

1029 which, further implies:

$$\left| (\hat{a} + \hat{b}) - (\hat{a}_0 + \hat{b}_0) \right| < \frac{\lambda_{2a}(1+p)\left(\ln\left(\frac{2p}{1-p}\right)\right)}{2p(1-p)}, \quad (61)$$

1030 from the same bound on \bar{g}' as before.

1031

Therefore, we showed that for any $0 < p < 3/4$, we have:

$$\left| \left(\hat{a} + \hat{b} \right) - \left(\hat{a}_0 + \hat{b}_0 \right) \right| < \frac{\lambda_{2a}(1+p) \left| \ln \left(\frac{2p}{1-p} \right) \right|}{2p(1-p)}. \quad (62)$$

1032

Combining this guarantee with the bound of (57) and the parameter space guarantee of (52), we finally have:

1033

$$\left| p_1 \left(\bar{\theta}_{2a} \middle| x_1 x_{d+1} = +1 \right) - \frac{1-p}{1+p} \right| < \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p) \left| \ln \left(\frac{2p}{1-p} \right) \right|}{8p(1-p)}. \quad (63)$$

Furthermore, for the optimality of $\bar{\theta}_{2a}$ in parameter space, we have:

$$\begin{aligned} & \left| \langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 \right| \\ &= \left| \langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b} - \hat{b} - \left(\langle \hat{a}_0 \mathbf{e}_{d+2} - \hat{a} \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \hat{b}_0 - \hat{b} \right) \right| \\ &\leq \sqrt{2(d+1)} \|\bar{\theta}_{2a} - \hat{\theta}\| + \left| \left(\hat{a}_0 + \hat{b}_0 \right) - \left(\hat{a} + \hat{b} \right) \right| \\ &< \frac{8(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p) \left| \ln \left(\frac{2p}{1-p} \right) \right|}{2p(1-p)}. \end{aligned} \quad (64)$$

where the last inequality follows from eqs. 62, 52. \square

As before, we obtain the following explicit corollary on the calibration and “hard” prediction of the second linear model.

Corollary 2. Consider running SGD (Algorithm 2) for minimizing $L_{\mathcal{D}_{2a}(p), \lambda_{2a}}(\mathbf{w}) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_{2a}(p)} [\ln(1 + e^{-y(\langle \mathbf{w}, \phi_2(\mathbf{x}) \rangle + b)})] + \frac{\lambda_{2a}}{2} (\|\mathbf{w}\|^2 + b^2)$. Then, for any $\varepsilon > 0$, after $T_{2a} = \tilde{O} \left(\max \left\{ \frac{d^2}{\delta \varepsilon^6}, \frac{d^2}{p^2 \delta \varepsilon^4} \right\} \right)$ iterations with $\lambda_{2a} = \tilde{\Theta} \left(\min \left\{ \frac{d^{\frac{1}{2}} p^{\frac{1}{2}}}{\delta^{1/4} T_{2a}^{1/4}}, \frac{d^{2/3}}{\delta^{1/3} T_{2a}^{1/3}} \right\} \right)$, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_{2a}} \sim \mathcal{D}_{2a}(p)$, it holds:

$$\begin{aligned} & \left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle - \bar{b}_{2a}}} - \frac{1-p}{1+p} \right| < \varepsilon, \forall \mathbf{x} \in \{\pm 1\}^{d+1} \text{ s.t. } x_1 x_{d+1} = +1, \\ & \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle + \bar{b}_{2a}}} < \varepsilon, \forall \mathbf{x} \in \{\pm 1\}^{d+1} \text{ s.t. } x_1 x_{d+1} = -1. \end{aligned} \quad (65)$$

If additionally $\varepsilon < \left| \ln \left(\frac{1-p}{2p} \right) \right|$, then with probability at least $1 - \delta$ we have:

$$\text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a}) = \text{sgn} \left(\ln \left(\frac{1-p}{2p} \right) \right) = \begin{cases} -1, & \text{if } p \geq 1/3, \\ +1, & \text{if } p < 1/3, \end{cases} \quad (66)$$

for all $\mathbf{x} \in \{\pm 1\}^{d+1}$ with $x_1 x_{d+1} = 1$.

Proof. Recall that from eqs. 46, 47, we have: for all $\mathbf{x} \in \{\pm 1\}^{d+1}$ with $x_1 x_{d+1} = -1$:

$$\begin{aligned} \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle + \bar{b}_{2a}}} &< \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \sqrt{\frac{\lambda_{2a}}{1-p}} \\ &< \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \sqrt{2\lambda_{2a}} := Q_a(\lambda_{2a}, T_{2a}), \end{aligned} \quad (67)$$

while for $\mathbf{x} \in \{\pm 1\}^{d+1}$ such that $x_1 x_{d+1} = +1$, it holds:

$$\begin{aligned} & \left| \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_{2a}, \mathbf{x} \rangle - \bar{b}_{2a}}} - \frac{1-p}{1+p} \right| < \frac{2(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p) \left| \ln \left(\frac{2p}{1-p} \right) \right|}{8p(1-p)}, \\ & \left| \langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 \right| < \underbrace{\frac{8(d+1)}{\lambda_{2a}} \sqrt{\frac{1 + \ln T_{2a}}{\delta T_{2a}}} + \frac{\lambda_{2a}(1+p) \left| \ln \left(\frac{2p}{1-p} \right) \right|}{2p(1-p)}}_{Q_b(\lambda_{2a}, T_{2a})}. \end{aligned} \quad (68)$$

1046 We want to upper bound all three quantities by $\varepsilon \in (0, 1)$. In (68), the second inequality upper
 1047 bounds the first one. Thus, it suffices to only consider that one. Let λ_a, λ_b be the optimal λ_{2a} 's for the
 1048 two previous expressions (i.e., quantities $Q_a(\lambda_{2a}, T_{2a}), Q_b(\lambda_{2a}, T_{2a})$) and T_a, T_b the corresponding
 1049 number of iterations to get the expressions less than ε . We have: $\lambda_a = \tilde{\Theta}\left(\frac{d^{2/3}}{\delta^{1/3}T_a^{1/3}}\right)$, while
 1050 $\lambda_b = \tilde{\Theta}\left(\frac{p^{1/2}d^{1/2}}{\delta^{1/4}T_b^{1/4}}\right)$ (by balancing the two terms of each of Q_a, Q_b). For these values of λ_a, λ_b , the
 1051 two expressions are equal to:

$$\begin{aligned} Q_a(\lambda_a, T_a) &= \tilde{\Theta}\left(\frac{d^{1/3}}{\delta^{1/6}T_a^{1/6}}\right), \\ Q_b(\lambda_b, T_b) &= \tilde{\Theta}\left(\frac{d^{1/2}}{p^{1/2}\delta^{1/4}T_b^{1/4}}\right). \end{aligned} \quad (69)$$

1052 This implies that it is sufficient to take $T_a = \tilde{O}\left(\frac{d^2}{\delta\varepsilon^6}\right)$ and $T_b = \tilde{O}\left(\frac{d^2}{p^2\delta\varepsilon^4}\right)$ to satisfy
 1053 $Q_a(\lambda_a, T_a) < \varepsilon$ and $Q_b(\lambda_b, T_b) < \varepsilon$, respectively. In other words, it is sufficient to take
 1054 $\lambda_{2a} = \tilde{\Theta}\left(\min\left\{\frac{d^{1/2}p^{1/2}}{\delta^{1/4}T_{2a}^{1/4}}, \frac{d^{2/3}}{\delta^{1/3}T_{2a}^{1/3}}\right\}\right)$ and $T_{2a} = \tilde{O}\left(\max\left\{\frac{d^2}{\delta\varepsilon^6}, \frac{d^2}{p^2\delta\varepsilon^4}\right\}\right)$. This proves the first
 1055 part of the claim. For the decision term $\text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + b_{2a})$, we have:

$$\begin{aligned} \text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + b_{2a}) &= \text{sgn}\left(\langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 + \langle \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \hat{b}_0\right) \\ &= \text{sgn}\left(\langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 + \hat{a}_0 x_1 x_{d+1} + \hat{b}_0\right). \end{aligned} \quad (70)$$

1056 In particular, when $x_1 x_{d+1} = 1$, we have:

$$\begin{aligned} \text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + b_{2a}) &= \text{sgn}\left(\langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 + \hat{a}_0 + \hat{b}_0\right) \\ &= \text{sgn}\left(\langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0 + \ln\left(\frac{1-p}{2p}\right)\right). \end{aligned} \quad (71)$$

1057 If $\left|\ln\left(\frac{1-p}{2p}\right)\right| > \left|\langle \bar{\mathbf{w}}_{2a} - \hat{a}_0 \mathbf{e}_{d+2}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a} - \hat{b}_0\right|$, then the following holds:
 1058 $\text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + b_{2a}) = \text{sgn}\left(\ln\left(\frac{1-p}{2p}\right)\right) = \text{sgn}(1/3 - p)$. For this to happen, it suf-
 1059 fices to additionally have $\varepsilon < \left|\ln\left(\frac{1-p}{2p}\right)\right|$.

1060 □

1061 **Position 2b** For the second part of position 2, we consider a binary classification problem where
 1062 all the data come from the “long” path. As this is similar to positions 3 to d, we treat all of these
 1063 positions together next. We will denote the second part of position 2 simply as position 2 in the next
 1064 subsection for ease of presentation.

1065 F.2.3 Positions 2 to d

1066 For the l 'th position of the output, $2 \leq l \leq d$, we consider a binary classification prob-
 1067 lem where $\mathcal{X}_l = \{\pm 1\}^{d+l-1}, \mathcal{Y}_l = \{\pm 1\}$ with a distribution $\mathcal{D}_l(p)$ over $\mathcal{X}_l \times \mathcal{Y}_l$ such that:
 1068 $x_1, \dots, x_d \sim \text{Rad}(1/2)$, $(x_{d+1}, \dots, x_{d+l-1}) = (x_1, \dots, \prod_{i=1}^{l-1} x_i)$ and $y = \prod_{i=1}^l x_i$. We con-
 1069 sider a linear hypothesis class $\mathcal{H}_l = \{\mathbf{x} \mapsto \langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle : \mathbf{w} \in \mathbb{R}^{2d+l-1}\}$, where the feature map
 1070 $\phi_l : \mathbf{x} \mapsto [x_1 \dots x_{d+l-1} \ x_{d+l-1}x_1 \dots x_{d+l-1}x_d]^T \in \{\pm 1\}^{2d+l-1}$ augments the input with
 1071 the second degree monomials that involve the inputs bits and the bit from the previous $(l-1)$ 'th
 1072 position. We consider learning with the logistic loss plus an additional ℓ_2 regularization term:
 1073 $l^{(l)}(\mathbf{w}, (\mathbf{x}, y)) = \ln(1 + e^{-y\langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle}) + \frac{\lambda_l}{2} \|\mathbf{w}\|^2$, $\lambda_l > 0$.

1074 **Proposition 3.** For any $l \in \{2, \dots, d\}$, consider running SGD (Algorithm 2) for minimizing
 1075 $L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w}) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_l(p)} [\ln(1 + e^{-y\langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle})] + \frac{\lambda_l}{2} \|\mathbf{w}\|^2$ with $\lambda_l > 0$. Then, after T_l it-
 1076 erations and for $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_l}$, it holds:

$$\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} < \frac{2d + l - 1}{2\lambda_l} \sqrt{\frac{1 + \ln T_l}{\delta T_l}} + \sqrt{\lambda_l}, \quad (72)$$

1077 for all $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ such that $x_{d+l-1} = \prod_{i=1}^{l-1} x_i \wedge \prod_{i=1}^l x_i = -1$, and

$$\frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} < \frac{2d + l - 1}{2\lambda_l} \sqrt{\frac{1 + \ln T_l}{\delta T_l}} + \sqrt{\lambda_l}, \quad (73)$$

1078 for all $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ such that $x_{d+l-1} = \prod_{i=1}^{l-1} x_i \wedge \prod_{i=1}^l x_i = 1$, where $\bar{\mathbf{w}}_l$ is the output of SGD.

1079 *Proof.* We first show that learning \mathcal{H}_l with respect to $l^{(l)}(\mathbf{w}, (\mathbf{x}, y)) = \ln(1 + e^{-y\langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle}) +$
 1080 $\frac{\lambda_l}{2} \|\mathbf{w}\|^2$ corresponds to a Convex and Lipschitz learning problem. The loss is λ_l -strongly convex
 1081 with respect to its first argument. For the Lipschitz constant, we have for all \mathbf{x}, y and $\mathbf{w} \in \mathbb{R}^{2d+l-1}$:

$$\left\| \nabla_{\mathbf{w}} \ln(1 + e^{-y\langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle}) \right\| = \left\| -\frac{y\phi_l(\mathbf{x})}{1 + e^{y\langle \mathbf{w}, \phi_l(\mathbf{x}) \rangle}} \right\| \leq \sqrt{2d + l - 1}. \quad (74)$$

1082 Therefore, applying Theorem 3, we have that SGD after T_l iterations returns a hypothesis $\bar{\mathbf{w}}_l$ such
 1083 that for any $\delta \in (0, 1)$ with probability at least $1 - \delta$ it holds:

$$L_{\mathcal{D}_l(p), \lambda_l}(\bar{\mathbf{w}}_l) \leq L_{\mathcal{D}_l(p), \lambda_l}(\hat{\mathbf{w}}_l) + \frac{2(2d + l - 1)}{\lambda_l \delta T_l} (1 + \ln T_l), \quad (75)$$

1084 where $\hat{\mathbf{w}}_l = \arg \min_{\mathbf{w} \in \mathcal{H}_l} L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w})$. From the strong convexity of $L_{\mathcal{D}_l(p), \lambda_l}$, this implies:

$$\|\bar{\mathbf{w}}_l - \hat{\mathbf{w}}_l\|^2 \leq \frac{2}{\lambda_l} (L_{\mathcal{D}_l(p), \lambda_l}(\bar{\mathbf{w}}_l) - L_{\mathcal{D}_l(p), \lambda_l}(\hat{\mathbf{w}}_l)) \leq \frac{4(2d + l - 1)}{\lambda_l^2 \delta T_l} (1 + \ln T_l). \quad (76)$$

1085 The previous bound on the parameter space can be translated to a guarantee on the calibration of the
 1086 model. First, we find the optimal solution $\hat{\mathbf{w}}_l = \arg \min_{\mathbf{w} \in \mathcal{H}_l} L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w})$. We set the gradient of
 1087 $L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w})$ to zero:

$$\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_l(p)} \left[\frac{-y\phi_l(\mathbf{x})}{1 + e^{y\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} \right] + \lambda_l \hat{\mathbf{w}}_l = 0. \quad (77)$$

1088 The objective $L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w})$ is strongly convex, so it admits a unique solution. We observe that this
 1089 solution is of the form $\hat{\mathbf{w}}_l = \alpha \mathbf{e}_{d+2l-1}$, $\alpha \in \mathbb{R}$, where $\mathbf{e}_{d+2l-1} = [0, \dots, 0, 1, 0, \dots, 0]^{T7}$. Indeed,
 1090 the optimality conditions become:

$$\begin{cases} \mathbb{E}_{x_1, \dots, x_d \sim \text{Rad}(1/2)} \left[\frac{-\prod_{i=1}^l x_i \prod_{i=1}^l x_i}{1 + e^{\alpha \prod_{i=1}^l x_i \prod_{i=1}^l x_i}} \right] + \lambda_l \alpha = 0, \\ \mathbb{E}_{x_1, \dots, x_d \sim \text{Rad}(1/2)} \left[\frac{-\phi_l(\mathbf{x})_j \prod_{i=1}^l x_i}{1 + e^{\alpha \prod_{i=1}^l x_i \prod_{i=1}^l x_i}} \right] = 0, j \neq d + 2l - 1. \end{cases} \quad (78)$$

1091 All but the $d + 2l - 1$ 'th equation are satisfied as $\mathbb{E}_{x_1, \dots, x_d \sim \text{Rad}(1/2)} \left[\prod_{i=1}^l x_i \phi_l(\mathbf{x})_j \right] = 0$ (there is
 1092 always at least one zero-mean bit that survives in the product). The $d + 2l - 1$ 'th equation can be
 1093 simplified to:

$$g(\alpha) := \frac{1}{1 + e^\alpha} - \lambda_l \alpha = 0. \quad (79)$$

⁷One way to “guess” the optimal solution is the following. Suppose the solution had an additional non-zero coefficient: $\hat{\mathbf{w}}_l = \alpha \mathbf{e}_{d+2l-1} + \beta \mathbf{e}_j$, $j \neq d + 2l - 1$. Take, for concreteness, $j = 1$ (the argument is invariant to the choice of j). Then, the optimality conditions yield: $\mathbb{E}_{\mathbf{x}} \left[\frac{-\prod_{i=1}^l x_i \prod_{i=1}^l x_i}{1 + e^{\alpha \prod_{i=1}^l x_i (\alpha \prod_{i=1}^l x_i + \beta x_1)}} \right] + \lambda_l \alpha = 0$,

and $\mathbb{E}_{\mathbf{x}} \left[\frac{-x_1 \prod_{i=1}^l x_i}{1 + e^{\alpha \prod_{i=1}^l x_i (\alpha \prod_{i=1}^l x_i + \beta x_1)}} \right] + \lambda_l \beta = 0$. By summing up and subtracting the equations, we get: $\frac{1}{1 + e^{\alpha + \beta}} - \lambda_l(\alpha + \beta) = 0$ and $\frac{1}{1 + e^{\alpha - \beta}} - \lambda_l(\alpha - \beta) = 0$. However, since the equation $\frac{1}{1 + e^u} - \lambda_l u = 0$ has a unique root for any $\lambda_l > 0$, it must be: $\alpha + \beta = \alpha - \beta$, or equivalently, $\beta = 0$.

1094 This equation has, indeed, a unique root as g is continuous and $g'(u) = -\frac{e^{-\alpha}}{(1+e^{-\alpha})^2} - \lambda_l < 0$ for
 1095 all $\lambda_l > 0, u \in \mathbb{R}$. Furthermore, $g(0) = 1 > 0$, while $g\left(\sqrt{\frac{1}{\lambda_l}}\right) = \frac{1}{1+e^{\frac{1}{\sqrt{\lambda_l}}}} - \sqrt{\lambda_l} < 0$, as for any
 1096 $\lambda_l > 0$ it holds: $1 + e^{\sqrt{\frac{1}{\lambda_l}}} \geq \sqrt{\frac{1}{\lambda_l}} + 2 > \sqrt{\frac{1}{\lambda_l}}$. From the intermediate value theorem, we get that
 1097 $\alpha \in \left(0, \sqrt{\frac{1}{\lambda}}\right)$. Hence, for the output probabilities of the optimal model $\hat{\mathbf{w}}_l$, we have:

$$\begin{aligned} p_1(\hat{\mathbf{w}}_l | \mathbf{x}) &= \sigma(\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \frac{1}{1 + e^{-\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} = \frac{1}{1 + e^{-\alpha \prod_{i=1}^l x_i}}, \\ p_{-1}(\hat{\mathbf{w}}_l | \mathbf{x}) &= \sigma(\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \frac{1}{1 + e^{\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} = \frac{1}{1 + e^{\alpha \prod_{i=1}^l x_i}}. \end{aligned} \quad (80)$$

1098 For any $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ with $x_{d+l-1} = \prod_{i=1}^{l-1} x_i$, the above together with eq. (79) imply:

$$\begin{aligned} p_1\left(\hat{\mathbf{w}}_l \middle| \prod_{i=1}^l x_i = -1\right) &= \frac{1}{1 + e^{\alpha}} = \lambda_l \alpha < \sqrt{\lambda_l}, \\ p_{-1}\left(\hat{\mathbf{w}}_l \middle| \prod_{i=1}^l x_i = 1\right) &= \frac{1}{1 + e^{\alpha}} = \lambda_l \alpha < \sqrt{\lambda_l} \end{aligned} \quad (81)$$

1099 Finally, combining with eq. (76), we have:

$$\begin{aligned} p_1\left(\bar{\mathbf{w}}_l \middle| \prod_{i=1}^l x_i = -1\right) &= \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} \\ &= \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} - \frac{1}{1 + e^{\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} + \frac{1}{1 + e^{\langle \hat{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} \\ &< \frac{\sqrt{2d+l-1}}{4} \|\bar{\mathbf{w}}_l - \hat{\mathbf{w}}_l\| + \sqrt{\lambda_l} \\ &\leq \frac{2d+l-1}{2\lambda_l} \sqrt{\frac{1+\ln T_l}{\delta T_l}} + \sqrt{\lambda_l}, \end{aligned} \quad (82)$$

1100 and, similarly:

$$p_{-1}\left(\bar{\mathbf{w}}_l \middle| \prod_{i=1}^l x_i = 1\right) < \frac{2d+l-1}{2\lambda_l} \sqrt{\frac{1+\ln T_l}{\delta T_l}} + \sqrt{\lambda_l}. \quad (83)$$

1101 □

1102 We obtain the following corollary on the calibration and “hard” prediction of the hypothesis returned
 1103 by SGD for the l ’th position of the output.

1104 **Corollary 3.** Consider running SGD (Algorithm 2) for minimizing $L_{\mathcal{D}_l(p), \lambda_l}(\mathbf{w}) =$
 1105 $\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_l(p)} [\ln(1 + e^{-y\langle \mathbf{w}, \mathbf{x} \rangle})] + \frac{\lambda_l}{2} \|\mathbf{w}\|^2$. Then, for any $\varepsilon > 0$, if $\lambda_l = \tilde{\Theta}\left(\frac{d^{2/3}}{\delta^{1/3} T_l^{1/3}}\right)$, and after
 1106 $T_l = \tilde{O}\left(\frac{d^2}{\delta \varepsilon^6}\right)$ iterations, with probability at least $1 - \delta$ over the sampled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{T_l} \sim \mathcal{D}_l(p)$, we
 1107 have:

$$\begin{aligned} \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \mathbf{x} \rangle}} &< \varepsilon, \forall \mathbf{x} \in \{\pm 1\}^{d+l-1} \text{ s.t. } x_{d+l-1} = \prod_{i=1}^{l-1} x_i \wedge \prod_{i=1}^l x_i = -1, \\ \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \mathbf{x} \rangle}} &< \varepsilon, \forall \mathbf{x} \in \{\pm 1\}^{d+l-1} \text{ s.t. } x_{d+l-1} = \prod_{i=1}^{l-1} x_i \wedge \prod_{i=1}^l x_i = 1. \end{aligned} \quad (84)$$

1108 If additionally, $\varepsilon < 1/2$, then with probability at least $1 - \delta$, we have:

$$\text{sgn}(\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \prod_{i=1}^l x_i, \quad (85)$$

1109 for all $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ with $x_{d+l-1} = \prod_{i=1}^{l-1} x_i$.

1110 *Proof.* Recall that from (72), we have:

$$\begin{aligned} \frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \mathbf{x} \rangle}} &< \frac{2d + l - 1}{2\lambda_l} \sqrt{\frac{1 + \ln T_l}{\delta T_l}} + \sqrt{\lambda_l}, \forall \mathbf{x} \in \{\pm 1\}^d \text{ s.t. } \prod_{i=1}^l x_i = -1, \\ \frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \mathbf{x} \rangle}} &< \frac{2d + l - 1}{2\lambda_l} \sqrt{\frac{1 + \ln T_l}{\delta T_l}} + \sqrt{\lambda_l}, \forall \mathbf{x} \in \{\pm 1\}^d \text{ s.t. } \prod_{i=1}^l x_i = 1, \end{aligned} \quad (86)$$

1111 Let $\lambda_l = \left(\frac{2d+l-1}{2}\right)^{2/3} \left(\frac{1+\ln T_l}{T_l}\right)^{1/3}$, then the upper bound becomes:

$$\frac{2d + l - 1}{2\lambda_l} \sqrt{\frac{1 + \ln T_l}{\delta T_l}} + \sqrt{\lambda_l} = 2 \left(\frac{2d + l - 1}{2}\right)^{1/3} \left(\frac{1 + \ln T_l}{\delta T_l}\right)^{1/6}. \quad (87)$$

1112 The first claim follows by solving the following inequality for T_l :

$$2 \left(\frac{2d + l - 1}{2}\right)^{1/3} \left(\frac{1 + \ln T_l}{\delta T_l}\right)^{1/6} < \varepsilon. \quad (88)$$

1113 For the second claim, observe that by a standard property of the logistic function, we have:

$$\text{sgn}(\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \text{sgn}\left(\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} - \frac{1}{2}\right). \quad (89)$$

1114 Assume that $\varepsilon < 1/2$, then we consider the cases:

1115 - If $\prod_{i=1}^l x_i = -1$, then we have:

$$\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} < \varepsilon < 1/2, \quad (90)$$

1116 which implies: $\text{sgn}\left(\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} - \frac{1}{2}\right) = -1$.

1117 - If $\prod_{i=1}^l x_i = 1$, then we have:

$$\frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} < \varepsilon < 1/2, \quad (91)$$

1118 which implies:

$$\begin{aligned} \text{sgn}\left(\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} - \frac{1}{2}\right) &= \text{sgn}\left(\frac{1}{1 + e^{-\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} - 1 + 1 - \frac{1}{2}\right) \\ &= \text{sgn}\left(-\frac{1}{1 + e^{\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle}} + \frac{1}{2}\right) \\ &= 1. \end{aligned} \quad (92)$$

1119 Therefore, for all $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ with $x_{d+l-1} = \prod_{i=1}^{l-1} x_i$, it holds:

$$\text{sgn}(\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \prod_{i=1}^l x_i. \quad (93)$$

1120 □

1121 F.2.4 End-to-end pre-training result

1122 **Theorem 4.** Let $d \geq 2$, $0 < \epsilon, \delta < 1$ and $0 < p < 3/4$. Let $\mathcal{D}(p)$ be a (parameterized)
 1123 distribution over sequences defined as in (18). Consider SGD (Algorithm 3) with per-position
 1124 iterations $T_1 = \tilde{O}\left(\frac{d^7}{\delta \epsilon^4}\right)$, $T_{2a} = \tilde{O}\left(\max\left\{\frac{d^9}{\delta \epsilon^6}, \frac{d^7}{p^2 \delta \epsilon^4}\right\}\right)$, $T_2, \dots, T_l = \tilde{O}\left(\frac{d^9}{\delta \epsilon^6}\right)$, total iter-
 1125 ations $T = \max\left\{T_1, T_{2a}, \frac{2 \max_{l \in \{2, \dots, d\}} T_l}{p}, \frac{8}{p} \ln \frac{2}{\delta}\right\}$ and with regularization coefficients $\lambda_1 =$

1126 $\tilde{\Theta}\left(\frac{d^{3/4}}{\delta^{1/4}T_1^{1/4}}\right), \lambda_{2a} = \tilde{\Theta}\left(\min\left\{\frac{d^{\frac{3}{4}}p^{\frac{1}{2}}}{\delta^{1/4}T_{2a}^{1/4}}, \frac{d}{\delta^{1/3}T_{2a}^{1/3}}\right\}\right), \lambda_l = \tilde{\Theta}\left(\frac{d}{\delta^{1/3}T_l^{1/3}}\right)$ for all $2 \leq l \leq d$.
 1127 Then, with probability at least $1 - \delta$, SGD returns hypotheses $\bar{\mathbf{w}}_1 \in \mathcal{H}_1, \bar{\boldsymbol{\theta}}_{2a} \in \mathcal{H}_{2a}, \bar{\mathbf{w}}_2 \in$
 1128 $\mathcal{H}_2, \dots, \bar{\mathbf{w}}_d \in \mathcal{H}_d$ that induce $h \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ for which for all $\mathbf{x} \in \{\pm 1\}^d$ it holds:

$$\begin{aligned} \left| \pi_h \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle EOS \rangle \right) \middle| \mathbf{x} \right) - p \right| &\lesssim \varepsilon, \\ \left| \pi_h \left(\left(\prod_{i=1}^d x_i, \langle EOS \rangle \right) \middle| \mathbf{x} \right) - \frac{1-p}{2} \right| &\lesssim \varepsilon. \end{aligned} \quad (94)$$

1129 Furthermore, if $\varepsilon < (d+1) \min \left\{ \frac{1}{2}, \ln \frac{1+p}{1-p}, \left| \ln \frac{1-p}{2p} \right| \right\}$, then it holds:

$$\hat{h}(\mathbf{x}; \{\bar{\mathbf{w}}_1, \bar{\boldsymbol{\theta}}_{2a}, \bar{\mathbf{w}}_2, \dots, \bar{\mathbf{w}}_d\}) = \begin{cases} (x_1, \langle EOS \rangle), & \text{if } p < 1/3, \\ (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle EOS \rangle), & \text{otherwise.} \end{cases} \quad (95)$$

1130 *Proof.* The proof strategy is to bound the probability of either not sampling enough ‘long’ sequences
 1131 or of SGD returning a ‘problematic’ hypothesis at some position. We invoke Corollaries 1 to 3 for
 1132 $\frac{\varepsilon}{d+1}, \frac{2}{2(d+1)}$. From Corollary 1, after $T \geq T_1 \left(\frac{\varepsilon}{d+1}, \frac{\delta}{2(d+1)} \right)$ iterations for $\lambda_1 = \tilde{\Theta} \left(\frac{d^{3/4}}{\delta^{1/4}T_1^{1/4}} \right)$,
 1133 SGD returns $\bar{\mathbf{w}}_1$ such that the following hold with probability less than $\frac{\delta}{2(d+1)}$:

$$\begin{aligned} \left| \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1 \right] - \frac{1+p}{2} \right| &\geq \frac{\varepsilon}{d+1}, \\ \left| \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \prod_{i=1}^d x_i \right] - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} \left(\mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = -x_1 \middle| \prod_{i=1}^d x_i = -x_1 \right] + \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1 \middle| \prod_{i=1}^d x_i = x_1 \right] \right) - \frac{1}{2} \right| \\ &\geq \frac{\varepsilon}{d+1}, \end{aligned} \quad (96)$$

1134 and, since $\frac{\varepsilon}{d+1} < \ln \frac{1+p}{1-p}$:

$$\hat{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle) = x_1. \quad (97)$$

1135 From Corollary 2, after $T \geq T_{2a} \left(\frac{\varepsilon}{d+1}, \frac{\delta}{2(d+1)} \right)$ iterations for $\lambda_{2a} =$
 1136 $\tilde{\Theta} \left(\min \left\{ \frac{d^{\frac{3}{4}}p^{\frac{1}{2}}}{\delta^{1/4}T_{2a}^{1/4}}, \frac{d}{\delta^{1/3}T_{2a}^{1/3}} \right\} \right)$, SGD returns $\bar{\mathbf{w}}_{2a}, \bar{b}_{2a}$ such that, it holds with probability
 1137 less than $\frac{\delta}{2(d+1)}$:

$$\left| \mathbb{P} \left[\tilde{h}_{2a}(\mathbf{x}; \bar{\boldsymbol{\theta}}_{2a}) = \epsilon \middle| \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1 \right] - \frac{2p}{1+p} \right| \geq \frac{\varepsilon}{d+1}, \quad (98)$$

1138 and, since $\frac{\varepsilon}{d+1} < \left| \ln \frac{1-p}{2p} \right|$:

$$\text{sgn}(\langle \bar{\mathbf{w}}_{2a}, \phi_2(\mathbf{x}) \rangle + \bar{b}_{2a}) = \text{sgn} \left(\ln \left(\frac{1-p}{2p} \right) \right) = \begin{cases} -1, & \text{if } p \geq 1/3, \\ +1, & \text{if } p < 1/3 \end{cases}, \quad (99)$$

1139 for all $\mathbf{x} \in \{\pm 1\}^{d+1}$ with $x_1 x_{d+1} = 1$.

1140 Let $z_1, \dots, z_T \sim \mathcal{D}(p)$ and let $T_{\text{long}} = \sum_{i=1}^T \mathbb{1} \{|z_i| > d+2\}$ be the count of ‘long’ samples. By
 1141 the definition of $\mathcal{D}(p)$, it is: $\mathbb{E}[T_{\text{long}}] = Tp$. By the multiplicative Chernoff bound, it holds:

$$\mathbb{P} \left[T_{\text{long}} \leq \frac{Tp}{2} \right] \leq e^{-\frac{Tp}{8}}, \quad (100)$$

1142 hence with probability at least $1 - \frac{\delta}{2}$, we have:

$$T_{\text{long}} > \frac{Tp}{2}, \quad (101)$$

1143 as long as $T > \frac{8}{p} \ln \frac{2}{\delta}$. If further, $T \geq \frac{2 \max_{l \in \{2, \dots, d\}} T_l}{p}$, then with probability at least $1 - \frac{\delta}{2}$:

1144 $T_{\text{long}} > T_l$ for all $l \in \{2, \dots, d\}$. From Corollary 3, after $T_{\text{long}} \geq T_l \left(\frac{\varepsilon}{d+1}, \frac{\delta}{2(d+1)} \right)$ iterations for

1145 $\lambda_l = \tilde{\Theta} \left(\frac{d}{\delta^{1/3} T_l^{1/3}} \right)$, SGD returns $\bar{\mathbf{w}}_l$ such that, with probability less than $\delta/2d$:

$$\left| \mathbb{P} \left[\tilde{h}_l(\mathbf{x}; \bar{\mathbf{w}}_l) = \prod_{i=1}^l x_i \mid \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1, \dots, \tilde{h}_{l-1}(\mathbf{x}; \bar{\mathbf{w}}_{l-1}) = x_{l-1} \right] - 1 \right| \geq \frac{\varepsilon}{d+1}, \quad (102)$$

1146 and, since $\frac{\varepsilon}{d+1} < \frac{1}{2}$:

$$\text{sgn}(\langle \bar{\mathbf{w}}_l, \phi_l(\mathbf{x}) \rangle) = \prod_{i=1}^l x_i \quad (103)$$

1147 for all $\mathbf{x} \in \{\pm 1\}^{d+l-1}$ with $x_{d+1} = \prod_{i=1}^{l-1} x_i$. Let E_{LONG} denote the event that $T_{\text{long}} > T_l$ for all
 1148 $2 \leq l \leq d$ and denote by P_i the event that the above guarantees hold for position i . Then, by union
 1149 bound, we have:

$$\begin{aligned} \mathbb{P}[E_{\text{LONG}} \wedge P_1 \wedge P_{2a} \wedge P_2 \wedge \dots P_d] &= \mathbb{P}[E_{\text{LONG}}] \mathbb{P}[P_1 \wedge P_{2a} \wedge P_2 \wedge \dots P_d | E_{\text{LONG}}] \\ &\geq \left(1 - \frac{\delta}{2}\right) (1 - \mathbb{P}[\exists i \in \{1, 2a, 2, \dots, d\} : \neg P_i]) \\ &\geq \left(1 - \frac{\delta}{2}\right) \left(1 - \sum_{i \in \{1, 2a, 2, \dots, d\}} \frac{\delta}{2(d+1)}\right) \geq 1 - \delta. \end{aligned} \quad (104)$$

1150 Thus, with probability at least $1 - \delta$, SGD returns a hypothesis h whose probability of correct, long
 1151 and short, sequences is:

$$\begin{aligned} \pi_h \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) &= \\ &= \mathbb{P}[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1] \cdot \dots \cdot \mathbb{P} \left[\tilde{h}_d(\mathbf{x}; \bar{\mathbf{w}}_d) = \prod_{i=1}^d x_i \mid \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1, \dots, \tilde{h}_{l-1}(\mathbf{x}; \bar{\mathbf{w}}_{l-1}) = x_{l-1} \right] \\ &= \left(\frac{1+p}{2} + \xi_1 \right) \left(\frac{2p}{1+p} + \xi_{2a} \right) (1 - \xi_2) \dots (1 - \xi_d), \end{aligned} \quad (105)$$

1152 and:

$$\begin{aligned} \pi_h \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) &= \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \prod_{i=1}^d x_i \right] \mathbb{P} \left[\tilde{h}_2(\mathbf{x}; \bar{\mathbf{w}}_2) = \langle \text{EOS} \rangle \mid \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \prod_{i=1}^d x_i \right] \\ &= \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \prod_{i=1}^d x_i \right] \left(\mathbb{P} \left[\tilde{h}_2(\mathbf{x}; \bar{\mathbf{w}}_2) = \langle \text{EOS} \rangle \mid \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1 \right] \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = x_1 \right] \right. \\ &\quad \left. + \mathbb{P} \left[\tilde{h}_2(\mathbf{x}; \bar{\mathbf{w}}_2) = \langle \text{EOS} \rangle \mid \tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = -x_1 \right] \mathbb{P} \left[\tilde{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = -x_1 \right] \right) \\ &= \left(\frac{1}{2} + \xi_1 \right) \left[\left(\frac{1-p}{1+p} + \xi_{2a} \right) \left(\frac{1+p}{2} + \xi_1 \right) + (1 - \xi_{2a}) \left(\frac{1-p}{2} - \xi_1 \right) \right], \end{aligned} \quad (106)$$

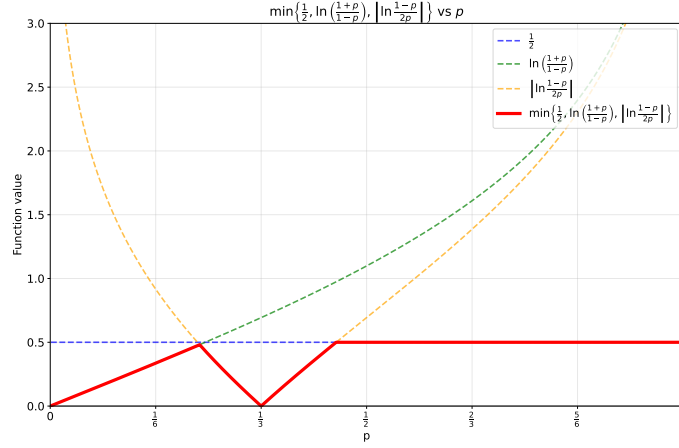


Figure 33: The functional form of the error term ε vs $p \in (0, 1)$ in Theorem 4.

1153 where $|\xi_1|, \dots, |\xi_d| \in O(\varepsilon/d)$. In other words,

$$\left| \pi_h \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - p \right| \lesssim \varepsilon. \quad (107)$$

1154 and

$$\left| \pi_h \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - \frac{1-p}{2} \right| \lesssim \varepsilon. \quad (108)$$

1155 Furthermore, from eqs. 97, 99, 103, for all $\mathbf{x} \in \{\pm 1\}^d$, we have:

1156 • $\hat{h}_1(\mathbf{x}; \bar{\mathbf{w}}_1) = \text{sgn}(\langle \bar{\mathbf{w}}_1, \mathbf{x} \rangle) = x_1,$

•

$$\begin{aligned} \text{sgn} \left(\left\langle \bar{\mathbf{w}}_{2a}, \phi_2 \left(\mathbf{x}, \hat{h}^{(1)}(\mathbf{x}; \bar{\mathbf{w}}_1) \right) \right\rangle + \bar{b}_{2a} \right) &= \text{sgn} \left(\ln \left(\frac{1-p}{2p} \right) \right) \\ &= \begin{cases} -1, & \text{if } 1/2 > p \geq 1/3, \\ +1, & \text{if } p < 1/3 \end{cases}, \end{aligned} \quad (109)$$

hence

$$\hat{h}_{2a}(\mathbf{x}; \bar{\boldsymbol{\theta}}_{2a}) = \begin{cases} \langle \text{EOS} \rangle, & \text{if } 1/2 > p \geq 1/3, \\ \epsilon, & \text{if } p < 1/3, \end{cases}$$

1157 and

1158 • $\hat{h}_l(\mathbf{x}; \bar{\mathbf{w}}_l) = \begin{cases} \prod_{i=1}^l x_i, & \text{if } \hat{h}_{2a}(\mathbf{x}; \bar{\boldsymbol{\theta}}_{2a}) \neq \langle \text{EOS} \rangle, \\ \epsilon, & \text{if } \hat{h}_{2a}(\mathbf{x}; \bar{\boldsymbol{\theta}}_{2a}) = \langle \text{EOS} \rangle, \end{cases} \quad \text{for } 2 \leq l \leq d.$

1159 This implies:

$$\hat{h}(\mathbf{x}; \{\bar{\mathbf{w}}_1, \bar{\boldsymbol{\theta}}_{2a}, \bar{\mathbf{w}}_2, \dots, \bar{\mathbf{w}}_d\}) = \begin{cases} (x_1, \langle \text{EOS} \rangle), & \text{if } p < 1/3, \\ (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle), & \text{otherwise.} \end{cases} \quad (110)$$

1160

□

1161 Theorem 1 in the main text follows as a corollary by Theorem 4 for $p = p_{\text{cot}}$. See Figure 33 for the
 1162 functional form of the error term $\varepsilon(p)$. In particular, the functional form also explains the difficulties
 1163 faced in training transformers for $p_{\text{cot}} \approx 1/3$.

1164 E.3 Post-training

1165 We complement the analysis of the previous subsection to show now a positive learning result on
 1166 the combined pre- & post-training procedure. Precisely, we show that, when $p_{\text{cot}} < 1/3$ which is
 1167 the hard regime for pre-training, only $O\left(\log \frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)$ RL rounds suffice for the pre-trained model to
 1168 reach perfect accuracy. Furthermore, we prove that the length of the answer grows in a predictable
 1169 way.

1170 **Theorem 5.** *Let $d \geq 2$, $\delta \in (0, 1)$ and $p_{\text{cot}} \in (0, 1/3)$. Let $\mathcal{D}(p_{\text{cot}})$ be a (parameterized) distribution
 1171 over sequences defined as in (18). Let $\varepsilon \leq c_0 \frac{p_{\text{cot}}}{1-p_{\text{cot}}}$ for some constant $c_0 > 0$. Consider pre-training
 1172 (Algorithm 3) per-position iterations $T_1 = \tilde{O}\left(\frac{d^7}{\delta \varepsilon^4}\right)$, $T_{2a} = \tilde{O}\left(\max\left\{\frac{d^9}{\delta \varepsilon^6}, \frac{d^7}{p_{\text{cot}}^2 \delta \varepsilon^4}\right\}\right)$, $T_2, \dots, T_l =$
 1173 $\tilde{O}\left(\frac{d^9}{\delta \varepsilon^6}\right)$, total iterations $T = \max\left\{T_1, T_{2a}, \frac{2 \max_{l \in \{2, \dots, d\}} T_l}{p}, \frac{8}{p_{\text{cot}}} \ln \frac{2}{\delta}\right\}$ and with regularization
 1174 coefficients $\lambda_1 = \tilde{\Theta}\left(\frac{d^{3/4}}{\delta^{1/4} T_1^{1/4}}\right)$, $\lambda_{2a} = \tilde{\Theta}\left(\min\left\{\frac{d^{3/4} p_{\text{cot}}^{1/2}}{\delta^{1/4} T_{2a}^{1/4}}, \frac{d}{\delta^{1/3} T_{2a}^{1/3}}\right\}\right)$, $\lambda_l = \tilde{\Theta}\left(\frac{d}{\delta^{1/3} T_l^{1/3}}\right)$ for
 1175 all $2 \leq l \leq d$. Furthermore, consider post-training with the STaR algorithm solving Problem LIN-RL
 1176 at each round and denote by $h^{(n)} \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ the model returned at the end of the n -th STaR round. Then,
 1177 there exists $n^* = O\left(\log \frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)$, such that, if:*

1178 1. SGD steps per-position per-STaR round are as follows: $T_1 = \tilde{O}\left(\frac{d^7}{\delta \varepsilon^4}\right)$,
 1179 $T_{2a} = \tilde{O}\left(\max\left\{\frac{d^9}{\delta \varepsilon^6}, \frac{d^7}{p_{\text{cot}}^2 \delta \varepsilon^4}\right\}\right)$, $T_2, \dots, T_l = \tilde{O}\left(\frac{d^9}{\delta \varepsilon^6}\right)$ and total iterations $T =$
 1180 $\max\left\{T_1, T_{2a}, \frac{2 \max_{l \in \{2, \dots, d\}} T_l}{p_{\text{cot}}}, \frac{8}{p_{\text{cot}}} \ln \frac{2 \log\left(\frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)}{\delta}\right\}$, and

1181 2. Regularization strengths per-position per-STaR round are as follows: $\lambda_1 = \tilde{\Theta}\left(\frac{d^{3/4}}{\delta^{1/4} T_1^{1/4}}\right)$,
 1182 $\lambda_{2a} = \tilde{\Theta}\left(\min\left\{\frac{d^{3/4} p_{\text{cot}}^{1/2}}{\delta^{1/4} T_{2a}^{1/4}}, \frac{d}{\delta^{1/3} T_{2a}^{1/3}}\right\}\right)$, $\lambda_l = \tilde{\Theta}\left(\frac{d}{\delta^{1/3} T_l^{1/3}}\right)$ for all $2 \leq l \leq d$,

1183 with probability at least $1 - \delta$ over sampling from $\mathcal{D}(p_{\text{cot}})$ (during pre-training) and over sam-
 1184 pling from models $\tilde{h}_0, \dots, \tilde{h}_{n^*}$ (during post-training), post-training returns $\bar{\mathbf{w}}_1^{(1)}, \dots, \bar{\mathbf{w}}_1^{(n^*)} \in$
 1185 $\mathcal{H}_1, \bar{\boldsymbol{\theta}}_{2a}^{(1)}, \dots, \bar{\boldsymbol{\theta}}_{2a}^{(n^*)} \in \mathcal{H}_{2a}, \bar{\mathbf{w}}_2^{(1)}, \dots, \bar{\mathbf{w}}_2^{(n^*)} \in \mathcal{H}_2, \dots, \bar{\mathbf{w}}_d^{(1)}, \dots, \bar{\mathbf{w}}_d^{(n^*)} \in \mathcal{H}_d$ that induce
 1186 $h^{(1)}, \dots, h^{(n^*)} \in \mathcal{H}_{\text{AR}}^{\text{Lin}} : \mathcal{X} \mapsto \mathcal{Y}$ for which for all $\mathbf{x} \in \mathcal{X}$ it holds:

$$\left| \pi_{h^{(n)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - q_n \right| \lesssim \varepsilon, \quad (111)$$

1187 where $|q_n - p_n| \lesssim 2^n \varepsilon$ and $p_{n+1} = \frac{2p_n}{1+p_n}$ for all $n \leq n^*$ with $p_0 = p_{\text{cot}}$, and also:

$$\hat{h}^{(n^*)} \left(\mathbf{x}; \left\{ \bar{\mathbf{w}}_1^{(n^*)}, \bar{\boldsymbol{\theta}}_{2a}^{(n^*)}, \bar{\mathbf{w}}_2^{(n^*)}, \dots, \bar{\mathbf{w}}_d^{(n^*)} \right\} \right) = \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right). \quad (112)$$

1188 *Proof.* The proof proceeds in two main steps. We first observe that the STaR objective for the
 1189 n 'th round can be re-written as next-token prediction with respect to a shifted version $\mathcal{D}(q_n)$ of the
 1190 original distribution $\mathcal{D}(p_{\text{cot}})$, where $0 < q_n < 1$. We then bound the deviation of sequence q_n from a
 1191 “noise-less” sequence p_n that describes the evolution of the proportion of long data in the data mix,
 1192 and whose closed form is available.

1193 From Theorem 4, for $p = p_{\text{cot}}$ and $\delta = \delta/2$, with probability at least $1 - \delta/2$, we obtain model
 1194 $h^{(0)} \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$, such that:

$$\left| \pi_{h^{(0)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - p_{\text{cot}} \right| \lesssim \varepsilon, \quad (113)$$

$$\left| \pi_{h^{(0)}} \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - \frac{1-p_{\text{cot}}}{2} \right| \lesssim \varepsilon,$$

1195 and

$$\hat{h}^{(0)}(\mathbf{x}; \{\bar{\mathbf{w}}_1, \bar{\boldsymbol{\theta}}_{2a}, \bar{\mathbf{w}}_2, \dots, \bar{\mathbf{w}}_d\}) = (x_1, \langle \text{EOS} \rangle). \quad (114)$$

1196 Recall the form of the STaR problem for the first round:

$$\min_{\mathbf{w}_1, \mathbf{w}_{2a}, b, \mathbf{w}_2, \dots, \mathbf{w}_d} \mathcal{L} \equiv \mathbb{E}_{\mathbf{x} \sim \text{Rad}(1/2)^{\otimes d}, y \sim \pi_{h^{(0)}}(\cdot | \mathbf{x})} [\mathbb{1}\{|y| = 2\} \mathcal{L}_{\text{short}} + \mathbb{1}\{|y| = d+1\} \mathcal{L}_{\text{long}} | r_{\text{cot}}(\mathbf{x}, y) = 1] \quad (115)$$

1197 where $\mathcal{L}_{\text{short}} = l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) + l^{(2a)}((\mathbf{w}_{2a}, b), ([\mathbf{x}, y_1], \tilde{y}_2))$ and $\mathcal{L}_{\text{long}} = l^{(1)}(\mathbf{w}_1, (\mathbf{x}, y_1)) +$
 1198 $l^{(2a)}(\{\mathbf{w}_{2a}, b\}, ((\mathbf{x}, y_1), \tilde{y}_2)) + \dots + l^{(d)}(\mathbf{w}_d, ((\mathbf{x}, y_1, \dots, y_{d-1}), y_d))$. Note that, with probability
 1199 at least $1 - \delta/2$, the objective can be re-written as:

$$\begin{aligned} \mathcal{L} &= \pi_{h^{(0)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x}, r_{\text{cot}}(\mathbf{x}, y) = 1 \right) \mathbb{E}_{\substack{x_1, \dots, x_d \sim \text{Rad}(1/2), \\ y = (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle)}} [\mathcal{L}_{\text{long}}] \\ &\quad + \pi_{h^{(0)}} \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x}, r_{\text{cot}}(\mathbf{x}, y) = 1 \right) \mathbb{E}_{\substack{x_1, \dots, x_d \sim \text{Rad}(1/2), \\ y = (\prod_{i=1}^d x_i, \langle \text{EOS} \rangle)}} [\mathcal{L}_{\text{short}}] \\ &= \frac{p_0 + \zeta_0}{\frac{p_0+1}{2} + \zeta_0 + \xi_0} \mathbb{E}_{\substack{x_1, \dots, x_d \sim \text{Rad}(1/2), \\ y = (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle)}} [\mathcal{L}_{\text{long}}] \\ &\quad + \left(1 - \frac{p_0 + \zeta_0}{\frac{p_0+1}{2} + \zeta_0 + \xi_0} \right) \mathbb{E}_{\substack{x_1, \dots, x_d \sim \text{Rad}(1/2), \\ y = (\prod_{i=1}^d x_i, \langle \text{EOS} \rangle)}} [\mathcal{L}_{\text{short}}] \\ &= \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}(q_1)} [\mathbb{1}\{|y| = 2\} \mathcal{L}_{\text{short}} + \mathbb{1}\{|y| = d+1\} \mathcal{L}_{\text{long}}], \end{aligned} \quad (116)$$

1200 where $q_1 := \frac{p_0 + \zeta_0}{\frac{p_0+1}{2} + \zeta_0 + \xi_0}$ with $|\zeta_0|, |\xi_0| \lesssim \varepsilon$. In other words, we showed that the reinforcement
 1201 learning risk with reward r_{cot} during the first round of STaR is equal to a next-token prediction risk,
 1202 where the distribution is the mixture of long and short strings but with shifted mixture weight q_1 .
 1203 Thus, from Theorem 4 for $p = q_1$ and $\delta = \delta_1 \in (0, 1)$ with probability $(1 - \delta_1)(1 - \delta/2)$, SGD
 1204 returns model $h^{(1)} \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$, such that:

$$\begin{aligned} \left| \pi_{h^{(1)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - q_1 \right| &\lesssim \varepsilon, \\ \left| \pi_{h^{(1)}} \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - \frac{1 - q_1}{2} \right| &\lesssim \varepsilon, \end{aligned} \quad (117)$$

1205 and

$$\hat{h}^{(1)}(\mathbf{x}; \{\bar{\mathbf{w}}_1, \bar{\boldsymbol{\theta}}_{2a}, \bar{\mathbf{w}}_2, \dots, \bar{\mathbf{w}}_d\}) = \begin{cases} (x_1, \langle \text{EOS} \rangle), & \text{if } q_1 < 1/3, \\ (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle), & \text{otherwise.} \end{cases} \quad (118)$$

1206 Therefore, by induction, as long as $q_n < 1/2$, invoking Theorem 1 n times for $\delta = \delta_i \in (0, 1)$
 1207 and $p = q_i$, with probability at least $\prod_{i=0}^n (1 - \delta_i) \geq 1 - \sum_{i=0}^n \delta_i$, post-training returns model
 1208 $h^{(n)} \in \mathcal{H}_{\text{AR}}^{\text{Lin}}$ such that:

$$\begin{aligned} \left| \pi_{h^{(n)}} \left(\left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - q_n \right| &\lesssim \varepsilon, \\ \left| \pi_{h^{(n)}} \left(\left(\prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right) \middle| \mathbf{x} \right) - \frac{1 - q_n}{2} \right| &\lesssim \varepsilon, \end{aligned} \quad (119)$$

1209 and

$$\hat{h}^{(n)}(\mathbf{x}; \{\bar{\mathbf{w}}_1, \bar{\boldsymbol{\theta}}_{2a}, \bar{\mathbf{w}}_2, \dots, \bar{\mathbf{w}}_d\}) = \begin{cases} (x_1, \langle \text{EOS} \rangle), & \text{if } q_n < 1/3, \\ (x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle), & \text{otherwise,} \end{cases} \quad (120)$$

1210 where $q_{n+1} = \frac{2(q_n + \zeta_n)}{q_n + 1 + 2(\zeta_n + \xi_n)}$ with $|\zeta_n|, |\xi_n| \lesssim \varepsilon$ with $q_0 = p_0$.

1211 We now analyze the perturbed sequence q_n . Let $f(q_n, \zeta_n, \xi_n) := q_{n+1} = \frac{2(q_n + \zeta_n)}{q_n + 1 + 2(\zeta_n + \xi_n)}$. We Taylor
1212 expand f around $(q_n, 0, 0)$:

$$\begin{aligned} f(q_n, \zeta_n, \xi_n) &= f(q_n, 0, 0) + \zeta_n \frac{\partial f}{\partial \zeta_n} \Big|_{(q_n, 0, 0)} + \xi_n \frac{\partial f}{\partial \xi_n} \Big|_{(q_n, 0, 0)} + R_f(\mathbf{u}, \zeta_n, \xi_n) \\ &= \frac{2q_n}{q_n + 1} + \zeta_n \frac{2 - 2q_n}{(q_n + 1)^2} + \xi_n \frac{-4q_n}{(q_n + 1)^2} + R_f(\mathbf{u}, \zeta_n, \xi_n) \\ &= \frac{2q_n}{q_n + 1} + \frac{2\zeta_n - 2\zeta_n q_n - 4\xi_n q_n}{(q_n + 1)^2} + R_f(\mathbf{u}, \zeta_n, \xi_n), \end{aligned} \quad (121)$$

1213 where we used the Lagrange form of the remainder: $R_f(\mathbf{u}, \zeta_n, \xi_n) = \frac{1}{2} H(0, \mathbf{u}) (\zeta_n^2 + \xi_n^2)$, where
1214 $\mathbf{u} \in \mathbb{R}^2$ such that: $|u_1| < |\zeta_n|$ and $|u_2| < |\xi_n|$ and $H(q, \zeta_n, \xi_n)$ is the Hessian of f evaluated at
1215 (q, ζ_n, ξ_n) . We now bound the deviation of the perturbed sequence q_{n+1} from the unperturbed one
1216 p_{n+1} :

$$\begin{aligned} |q_{n+1} - p_{n+1}| &= \left| f(q_n, \zeta_n, \xi_n) - \frac{2p_n}{p_n + 1} \right| \\ &= \left| \frac{2q_n}{q_n + 1} + \frac{2\zeta_n - 2\zeta_n q_n - 4\xi_n q_n}{(q_n + 1)^2} + R_f(\mathbf{u}, \zeta_n, \xi_n) - \frac{2p_n}{p_n + 1} \right| \\ &\leq \left| \frac{2q_n}{q_n + 1} - \frac{2p_n}{p_n + 1} \right| + \left| \frac{2\zeta_n - 2\zeta_n q_n - 4\xi_n q_n}{(q_n + 1)^2} \right| + |R_f(\mathbf{u}, \zeta_n, \xi_n)| \\ &\leq 2|q_n - p_n| + \left| \frac{2\zeta_n - 2\zeta_n q_n - 4\xi_n q_n}{(q_n + 1)^2} \right| + |R_f(\mathbf{u}, \zeta_n, \xi_n)|, \text{ provided } q_n > 0. \end{aligned} \quad (122)$$

1217 Since $q_n < 1$ and $|\zeta_n|, |\xi_n| \in O(\varepsilon)$ and $|R_f(\mathbf{u}, \zeta_n, \xi_n)| \lesssim \varepsilon^2$, if we denote the error sequence by
1218 $e_n = |q_n - p_n|$, then:

$$e_{n+1} - 2e_n \lesssim \varepsilon. \quad (123)$$

1219 Since $e_0 = 0$, this implies that the error at step n is bounded as:

$$e_n \lesssim 2^n \varepsilon. \quad (124)$$

1220 We want to find an integer n^* such that model $h^{(n^*)}$ generates long answers. For this, it suffices to
1221 have $q_n > 1/3$, or equivalently:

$$p_{n^*} - 2^{n^*} C' \varepsilon > \frac{1}{3}. \quad (125)$$

1222 We show that the unperturbed sequence $p_n = \frac{2p_{n-1}}{p_{n-1} + 1}$ admits a closed form solution. Let $u_n = \frac{1}{p_n}$,
1223 then:

$$u_n = \frac{u_{n-1} + 1}{2} = 1 + \frac{u_0 - 1}{2^n}, \quad (126)$$

1224 or, for the original sequence p_n :

$$p_n = \left(1 + \frac{1}{2^n} \left(\frac{1}{p_0} - 1 \right) \right)^{-1} = \frac{2^n p_0}{2^n p_0 + 1 - p_0} = \frac{p_0}{p_0 + (1 - p_0) 2^{-n}}. \quad (127)$$

1225 Returning back to (125), we seek $n^* \geq 1$ such that:

$$\frac{p_0}{p_0 + (1 - p_0) 2^{-n^*}} - 2^{n^*} C' \varepsilon > \frac{1}{3}. \quad (128)$$

1226 Assume that $2^{n^*} C' \varepsilon < \frac{1}{15}$ for some ε to be specified later. Then, it suffices to have:

$$\frac{p_0}{p_0 + (1 - p_0) 2^{-n^*}} - \frac{1}{15} > \frac{1}{3}, \quad (129)$$

1227 or

$$n^* > \log \left(\frac{2(1-p_0)}{3p_0} \right). \quad (130)$$

1228 In that case, the error parameter ε needs to be as small as:

$$\varepsilon < \frac{1}{C} \frac{p_0}{1-p_0}, \quad (131)$$

1229 for some $C > 0$. Finally, we pick the smallest n^* such that $q_{n^*} > 1/3 + \mu$, where $\mu \geq \omega(e^{-d})^8$.
 1230 Note that $p_n < 1/3 \implies p_{n+1} < 1/2$, hence Theorem 4 can be invoked for $q_{n^*} < 3/4$ if C is
 1231 sufficiently small. We set $\delta_0 = \delta/2$ and $\delta_1, \dots, \delta_{n^*} = \frac{\delta}{2^{n^*}}^9$, then with probability at least $1 - \delta$, we
 1232 have:

$$\hat{h}^{(n^*)} \left(\mathbf{x}; \left\{ \bar{\mathbf{w}}_1^{(n^*)}, \bar{\boldsymbol{\theta}}_{2a}^{(n^*)}, \bar{\mathbf{w}}_2^{(n^*)}, \dots, \bar{\mathbf{w}}_d^{(n^*)} \right\} \right) = \left(x_1, x_1 x_2, \dots, \prod_{i=1}^d x_i, \langle \text{EOS} \rangle \right). \quad (132)$$

1233

□

1234 *Remark 5.* Note that the guarantees of Theorem 5 for the number of SGD iterations required per
 1235 STaR round are very pessimistic: 1) they depend on the original mixture weight p_{cot} rather than the
 1236 per-round updated weight q_n , and 2) they assume that the models are re-initialized at the origin prior
 1237 to each STaR round, whereas of course in practice we continue fine-tuning the previously obtained
 1238 model which speeds up convergence.

1239 *Remark 6.* One can also analyze an (almost offline) version of REINFORCE algorithm with no
 1240 reward normalization, instead of STaR, by noting that its loss also corresponds to the next-token
 1241 prediction objective but scaled by a different per-round coefficient.

1242 In particular, if the proportion of long data in the data mixture is not exponentially small, i.e., there
 1243 exists constant $\kappa \in \mathbb{N}$ such that: $p_{\text{cot}} \in \Omega(d^{-\kappa})$, then by following the previous pre- & post-training
 1244 recipe on data coming from $\mathcal{D}(p_{\text{cot}})$, we can obtain a model that emits a long, correct sequence and
 1245 predicts the parity of d bits after $O(\text{poly}(d))$ SGD iterations.

⁸We require q_{n^*} to not be exponentially close to $1/3$, so that we do not pay $1/\varepsilon$ sample complexity for
 $\varepsilon < (d+1) \ln \frac{2q_{n^*}}{1-q_{n^*}} \asymp (d+1)(q_{n^*} - 1/3)$

⁹Note that for $\delta_i = \frac{\delta}{2^{n^*}}$ the regularization terms and SGD iterations grow together with a factor
 $O\left(\ln \frac{1-p_{\text{cot}}}{p_{\text{cot}}}\right)$, but this gets suppressed by the $\tilde{O}(\cdot)$ notation.