PRETRAINING WITH MASKED BACKSTORIES IN A TOY WORLD

Anonymous authors
Paper under double-blind review

ABSTRACT

Context-enhanced learning (CEL) involves augmenting context in large language models (LLMs) with special masked context to accelerate learning. It has previously only been explored using LLMs with billions of parameters during finetuning because CEL requires in-context-learning (ICL) abilities to work. Here, we leverage a toy world (symbolically labeled randomly interleaved vector timeseries from linear deterministic dynamical systems) that admits LLM-style "next token" pretraining and has been shown to exhibit multiple emergences of different ICL/recall abilities in tiny transformer models with mere millions of parameters.

In this toy world, we can see a late transition from ICL to in-weights learning that also corresponds to a degradation of ICL performance on time-series from systems never seen during training. We enhance pretraining with additional masked context that allows the model to make near perfect predictions on the original training examples. Masking this additional context disincentivizes the model from memorizing it, and the capability of perfect prediction on the training example disincentivizes the model from memorizing the remaining portion of the training example. Not only does this enhancement suppress in-weights learning of the specific training systems, but we also show that it improves the quality of ICL in the model, including the seemingly unrelated task of associative recall. Even more surprisingly, another experiment shows that despite such a model during training only seeing losses (and hence gradients) for tokens that are perfectly predictable, it generalizes well at test time when predicting tokens that are not perfectly predictable, nearly matching the performance of the optimal solution for those cases.

1 INTRODUCTION

The machine learning community is used to the general idea that models get better with scale. We carve out an exception for "overfitting/overtraining" but reality is more complicated. The phenomenon of "inverse scaling" exists (Lin et al., 2022; Michaelov & Bergen, 2023) and McKenzie et al. (2024) showed that some capabilities of language models may get worse (like subsequent finetunability as in Springer et al. (2025)) as model size and training compute grows[1], although many other capabilities are getting better[2]. An example of an inverse scaling task is prompting a model to always complete with the word "heavy" and then giving it "Absence makes the heart grow " — larger models are more likely to disregard the instructions and respond "fonder." This classic example illustrates a general pattern — inverse-scaling examples seem to manifest a tension between what's been learned in the weights (IWL) and what should be learned from the context (ICL).

In this paper, we study an algorithmic toy problem, where small scale models (millions of parameters) exhibit both IWL and ICL *as well as* the phenomenon of some capabilities getting better while others get worse at some point in training. We investigate a technique for modulating the develop-

[1] Wei et al. (2023); Wu & Lo (2024) add further nuance to the inverse scaling discussion, by showing that at even larger scales, there can be "double descent" (Nakkiran et al., 2021) improvement even after some worsening in the inverse-scaling capabilities.

[2] This complex dynamics of the LLM capability evolution necessitates a large repertoire of methods for evaluating, controlling, and modulating the progress of language models at a finer-grained capability by capability level. Currently, model merging, data curation and context distillation, and post-training techniques are some methods that try to mitigate this problem. See Appendix A for more.

ment of a capability during training inspired by context-enhanced learning (Zhu et al., 2025) and gradient starvation (Tachet et al., 2020; Pezeshki et al., 2021). The approach novelty in our work is twofold: (a) Our toy problem exhibits the "some abilities get better while others get worse" during training — earlier works studying ICL/IWL transitions did not have toys that showed both directions simultaneously; and (b) we engage with an intervention of "masked context" which is qualitatively and substantively distinct from the kinds of regularization or data-mix curation interventions studied in earlier toys. We show that the intervention works in blocking the transition to IWL, ends up strengthening ICL including a seemingly unrelated aspect of associative recall, and even more surprisingly, suggests that there is something about the inductive bias of these architectures that lets them successfully extrapolate to a task that they've never seen training gradients for.

2 SETUP

Building off of Garg et al. (2022); Sander et al. (2024), we focus on the orthogonally evolved system family, where the system is defined by $U \in \mathbb{R}^{5\times5}$, a random orthogonal matrix. Each U is generated (Mezzadri, 2006) to ensure a uniform sampling over all $\mathbb{R}^{5\times5}$ orthogonal matrices. The initial state is $\mathbf{x}_0 \sim \mathcal{N}\left(0, \frac{1}{5}I\right)$, with state updates: $\mathbf{x}_{i+1} = U\mathbf{x}_i = U^{i+1}\mathbf{x}_0$. The system state is in-principle perfectly predictable, but only after six positions are observed:

$$U = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5\end{bmatrix}\begin{bmatrix}\mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4\end{bmatrix}^{-1}. \tag{1}$$

An optimal algorithm (for mean squared error) for learning the underlying system from state observations is to use the pseudoinverse in place of the inverse as presented in Appendix B.1.

Following Arora et al. (2023), we add an aspect of associative recall to our toy task. To form a training trace, we interleave segments of observation sequences from a library of 40,000 orthogonal systems into a length-251 context window. The exact interleaving procedure is described in Appendix B.2 and illustrated in Fig. 1.

Figure 1: Example interleaved training example. The different colors correspond to the different segments that are being interleaved. The purple sequence is generated from system 30 in the training library (denoted in the superscript of the initial state and the subscript of the U matrix), while the blue sequence is generated from system 2. Notice that each system in the training trace has its own pair of symbolic punctuation labels (SPLs) represented by parentheses, curly braces, and brackets in the figure. Notice that when the same system appears again, the current sequence continues from where the previous sequence ended, as seen by the last purple segment starting from index 25 since the first purple segment ended at index 24, and the last blue segment starting at index 5 since the first blue segment ended at index 4.

2.1 NEEDLE-IN-A-HAYSTACK TESTING

For testing, we generate 100 held-out systems and 1000 different held-out initial states by the same method described in Section B.2 to form our *testing library*. Since the testing systems do not appear in the training set, if the model is able to achieve low error on testing systems, then it must be able to in-context learn new orthogonal systems. For testing, we also include N distinct systems with 10 state observations each into one "needle-in-a-haystack" trace, followed by the continuation of one of them to test associative recall.

A more detailed version of this procedure is in Appendix B.3. See Fig. 8 for a diagram of a test trace for $N = 2$ systems in the haystack and system U_1 as the needle.

3 TRAINING WITH MASKED BACKSTORIES

Whenever any particular system appears in a training example for the first time, it begins with the idiosyncratic initial state that happened to be randomly sampled for that system when building the

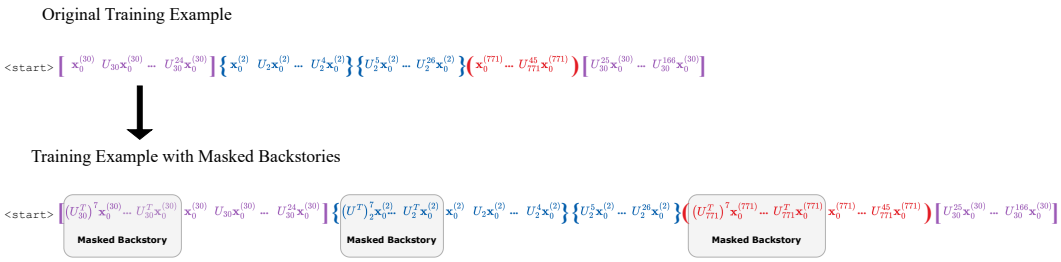
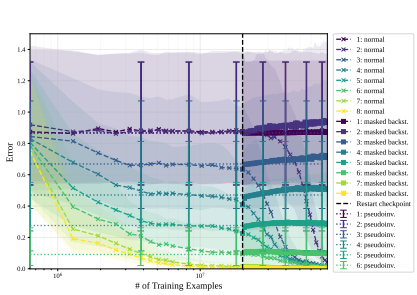
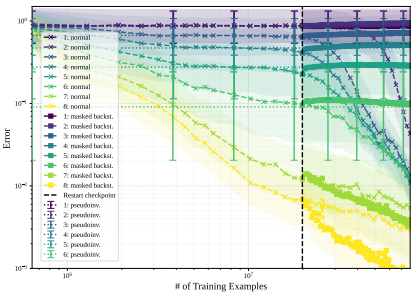


Figure 2: Training with masked backstories consists of reversing the evolution of a linear system when it appears in the training example for the first time, prepending these state observations to the corresponding segment, and masking any training loss on these backstory indices.

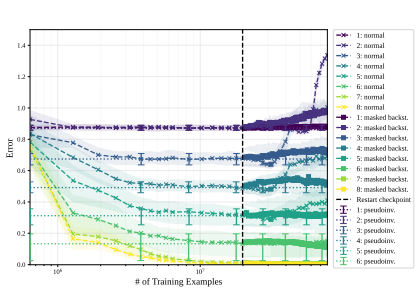
training library as specified in step 9 in Section B.2. These early indices (from the first to the sixth) are not perfectly predictable by pure ICL alone, since 6 state observations are required to predict with an MSE of 0 as shown in (1). Looking at Figs. 3a and 3b which show the performance on training traces, the crossed curves dip below their corresponding dotted lines towards zero squared error after more than 1×10^7 training examples have been seen by the model during training. The only way to achieve this preternatural performance is by memorizing facts rather than doing ICL.



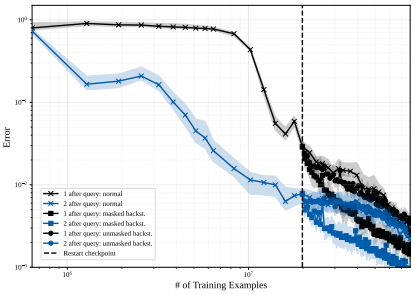
(a) Training traces. Linear scale.



(b) Training traces. Log scale.



(c) Test traces. Linear scale. Initial segment.



(d) Test traces. Log scale. Test segment.

Figure 3: The median squared error of the normally trained model (crossed curves), the masked backstory trained model (squared curves), and the pseudoinverse predictor (dotted lines). The horizontal axes measure training progress in terms of examples seen. (Multiply by 250 to get tokens)

In Fig. 3c, notice that after seeing 3×10^7 training examples, the model's error on held-out test data sharply increases above the level of the corresponding pseudoinverse predictor levels. The transition to IWL has hurt ICL performance.

To suppress this in-weights learning (in the style of Lampinen et al. (2025); Park et al. (2025b)), if a system U appears for the first time in a training example, we evolve the system in reverse for seven time steps by successively applying U^T to the initial state $\mathbf{x}_0$ to obtain $\{\mathbf{x}_{-7}, \ldots, \mathbf{x}_{-1}\}$. We call these 7 state observations a "backstory". We then prepend this backstory to the system segment as seen in Fig. 2. Finally, when the training loss is computed, we mask any losses incurred on the

backstory indices. This masking is qualitatively identical to the CEL approach in Zhu et al. (2025), and so is in the spirit of leaning on the model's ICL ability.

3.1 Restarting Training with Masked Backstories

Our main experiment is to choose a checkpoint (at 2×10^7 training examples up to that point in training, and we call this checkpoint the "restart checkpoint") before the generalization performance has deteriorated and continue training with masked backstories from that point. We also conducted experiments where a model was always trained with masked backstories, and never received loss on any of first 7 indices of an initial segment. This means neither IWL nor ICL were ever encouraged for these indices. Appendix C shows that these models *still* manage to make predictions that converge towards optimal on indices that were never lossed.

Fig. 3a shows that on the training data, the intervention by masked backstories completely blocks and even reverses the memorization of the training data. For the most part, the curves are near the level of the optimal psuedo-inverse predictor, with some additional degradation visible for the earlier time steps. The transition to IWL appears blocked, and indeed Fig. 3c shows that on held-out test traces, the degradation in ICL performance seen in the crossed curves is largely not present in the square curves with the masked-backstory intervention. Instead, we only see the same slight degradation in the purple and blue curves corresponding to earlier time steps that we see in the training data. There's no discrepancy between the training and test.

Log-scale plots in Fig 3b and Fig 3d show a further benefit from the intervention. In Fig 3b we see that the performance for later time indices (when perfect ICL reconstruction is possible) gets even better with the masked backstory intervention (the Appendix shows this is the same with held-out data). And Fig. 3d shows that the black (and blue) curves associated with associative recall performance continue to improve, and actually improve even faster with the masked backstories.

4 Discussion

Training models to perform next-token prediction on interleaved time-series generated from linear dynamical systems provides a controlled setting for studying training dynamics. The combination of associative recall (from interleaving) and least-squares-style ICL (from the dynamics) allowed for the toy to exhibit multiple capabilities, in which an ICL-IWL transition hurt one of them while another continued to improve during training. This simple data model also facilitated the conception of training with masked backstories, and devising experiments to validate this technique.

The observation that models trained with masked backstories throughout the entirety of training (Appendix C) are still able to make reasonable predictions for indices that were always masked opens up questions about the inductive biases of the transformer architecture. There are infinitely many possible predictors for this problem that would achieve zero MSE once six observations are seen, but would obtain worse MSE than the pseudoinverse predictor when predicting the first six indices. For example, randomly projecting each observation vector, performing the pseudoinverse prediction in the new subspace, and then projecting the predicted vector back to the original subspace. An interesting theoretical question is, why does the transformer architecture have the proclivity to find something close to the min-norm solution in original coordinates when the training objective does not require this and weight matrices are initialized randomly? Empirically, are there other tasks or scenarios where a model can incidentally learn a capability without any loss fueling its learning?

Furthermore, it is unclear exactly why the additional masked context leads to improved in-context associative recall and in-context learning on later indices in initial segments. This phenomenon closely aligns with the claims of Singh et al. (2025) where ICL and "context-constrained" IWL compete in the first attention layer of a two-layer transformer, but share circuitry in the second layer. Suppressing IWL with masked backstories could then be allowing the ICL circuit to develop stronger in the early layers of our model without the interference of IWL circuit development.

As Qi et al. (2025) showed that it is possible to automatically generate additionally context for language data using another LLM (in their case GPT-4), this opens the opportunity for larger-scale experiments for language model training with masked backstories.

REFERENCES

Suraj Anand, Michael A. Lepori, Jack Merullo, and Ellie Pavlick. Dual process learning: Controlling use of in-context vs. in-weights strategies with weight forgetting, 2025. URL https://arxiv.org/abs/2406.00053.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

Per Bak. *How nature works : the science of self-organized criticality*. Copernicus, New York, NY, USA, 1996. ISBN 9780387987385.

Bryan Chan, Xinyi Chen, András György, and Dale Schuurmans. Toward understanding in-context vs. in-weight learning, 2025. URL https://arxiv.org/abs/2410.23042.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in neural information processing systems*, 35:18878–18891, 2022.

Yihong Chen, Kelly Marchisio, Roberta Raileanu, David Adelani, Pontus Lars Erik Saito Stenetorp, Sebastian Riedel, and Mikel Artetxe. Improving language plasticity via pretraining with active forgetting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31543–31557. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6450ea28ebbc8437bc38775157818172-Paper-Conference.pdf.

Jasper Dekoninck, Marc Fischer, Luca Beurer-Kellner, and Martin Vechev. Controlled text generation via language model arithmetic, 2024. URL https://arxiv.org/abs/2311.14479.

Zhe Du, Haldun Balim, Samet Oymak, and Necmiye Ozay. Can transformers learn optimal filtering for unknown systems? *IEEE Control Systems Letters*, 7:3525–3530, 2023. doi: 10.1109/LCSYS.2023.3335318.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegreffe. The unreasonable effectiveness of easy training data for hard tasks, 2024. URL https://arxiv.org/abs/2401.06751.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Alexander Y. Ku, Thomas L. Griffiths, and Stephanie C. Y. Chan. Predictability shapes adaptation: An evolutionary perspective on modes of learning in transformers, 2025. URL https://arxiv.org/abs/2505.09855.

Andrew K Lampinen, Arslan Chaudhry, Stephanie CY Chan, Cody Wild, Diane Wan, Alex Ku, Jörg Bornschein, Razvan Pascanu, Murray Shanahan, and James L McClelland. On the generalization of language models from in-context learning and finetuning: a controlled study. *arXiv preprint arXiv:2505.00661*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Shalini Maiti, Amar Budhiraja, Bhavul Gauri, Gaurav Chaurasia, Anton Protopopov, Alexis Audran-Reiss, Michael Slater, Despoina Magka, Tatiana Shavrina, Roberta Raileanu, and Yoram Bachrach. Souper-model: How simple arithmetic unlocks state-of-the-art llm performance, 2025. URL https://arxiv.org/abs/2511.13254.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Ian R. McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, Andrew Gritsevskiy, Daniel Wurgaft, Derik Kauffman, Gabriel Recchia, Jiacheng Liu, Joe Cavanagh, Max Weiss, Sicong Huang, The Floating Droid, Tom Tseng, Tomasz Korbak, Xudong Shen, Yuhui Zhang, Zhengping Zhou, Najoung Kim, Samuel R. Bowman, and Ethan Perez. Inverse scaling: When bigger isn't better, 2024. URL https://arxiv.org/abs/2306.09479.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006.

James Michaelov and Ben Bergen. Emergent inabilities? inverse scaling over the course of pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14607–14615. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.973. URL http://dx.doi.org/10.18653/v1/2023.findings-emnlp.973.

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36, 2023.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning, 2024. URL https://arxiv.org/abs/2412.00104.

Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. Competition dynamics shape algorithmic phases of in-context learning, 2025a. URL https://arxiv.org/abs/2412.01003.

Core Francisco Park, Zechen Zhang, and Hidenori Tanaka. New news: System-2 fine-tuning for robust integration of new knowledge. *arXiv preprint arXiv:2505.01812*, 2025b.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks, 2021. URL https://arxiv.org/abs/2011.09468.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions, 2025. URL https://arxiv.org/abs/2406.11194.

Neil Rathi and Alec Radford. Shaping capabilities with token-level data filtering, 2026. URL https://arxiv.org/abs/2601.21571.

Yeonju Ro, Cong Xu, Agnieszka Ciborowska, Suparna Bhattacharya, Frankie Li, and Martin Foltin. Dataset efficient training with model ensembling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4700–4704, June 2023.

Michael Eli Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. How do transformers perform in-context autoregressive learning ? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43235–43254. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/sander24a.html.

Hinrich Schutze and Christopher Manning. I preliminaries. In *Foundations of Statistical Natural Language Processing*. MIT Press, United States, 1999. ISBN 9780262133609.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3501672ebc68a5524629080e3ef60aef-Paper.pdf.

Thibault Sellam, Steve Yadlowsky, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, Ian Tenney, and Ellie Pavlick. The multiberts: Bert reproductions for robustness analysis, 2022. URL https://arxiv.org/abs/2106.16163.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Aaditya Singh, Stephanie Chan, Ted Moskovitz, Erin Grant, Andrew Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 36, 2023.

Aaditya K. Singh, Ted Moskovitz, Sara Dragutinovic, Felix Hill, Stephanie C. Y. Chan, and Andrew M. Saxe. Strategy coopetition explains the emergence and transience of in-context learning, 2025. URL https://arxiv.org/abs/2503.05631.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. URL https://arxiv.org/abs/2209.15189.

Jiajun Song, Zhuoyan Xu, and Yiqiao Zhong. Out-of-distribution generalization via composition: A lens through induction heads in transformers. *Proceedings of the National Academy of Sciences*, 122(6):e2417182122, 2025. doi: 10.1073/pnas.2417182122. URL https://www.pnas.org/doi/abs/10.1073/pnas.2417182122.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune, 2025. URL https://arxiv.org/abs/2503.19206.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 51118–51168. Curran Associates, Inc., 2024. doi: 10.52202/079017-1618. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5b6346a05a537d4cdb2f50323452a9fe-Paper-Conference.pdf.

Marton Szep, Jorge Marin Ruiz, Georgios Kaissis, Paulina Seidl, Rüdiger von Eisenhart-Rothe, Florian Hinterwimmer, and Daniel Rueckert. Unintended memorization of sensitive information in fine-tuned language models, 2026. URL https://arxiv.org/abs/2601.17480.

Remi Tachet, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. On the learning dynamics of deep neural networks, 2020. URL https://arxiv.org/abs/1809.06848.

Jason Wei, Najoung Kim, Yi Tay, and Quoc Le. Inverse scaling can become U-shaped. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15580–15591, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.963. URL https://aclanthology.org/2023.emnlp-main.963/.

Tung-Yu Wu and Pei-Yu Lo. U-shaped and inverted-u scaling behind emergent abilities of large language models. *ArXiv*, abs/2410.01692, 2024. URL https://api.semanticscholar.org/CorpusID:273025558.

Alexander Xiong, Xuandong Zhao, Aneesh Pappu, and Dawn Song. The landscape of memorization in llms: Mechanisms, measurement, and mitigation, 2025. URL https://arxiv.org/abs/2507.05578.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 2024.

Xingyu Zhu, Abhishek Panigrahi, and Sanjeev Arora. On the Power of Context-Enhanced Learning in LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

A Related Work

Memorization in LLMs It has been observed that modern large language models sometimes memorize training examples into their weights (Xiong et al., 2025; Szep et al., 2026). This memorization is a strong instance of in-weights learning (IWL), where models make predictions based on weight updates Chan et al. (2022); Singh et al. (2023). Still, this memorization may help a language model generate factually correct text with good grammatical structure.

Table 1: Toy models of in-weights and in-context learning dynamics

Paper	Task Dataset	Transformer Architecture	Claims
Chan et al. (2022)	Omniglot	12 layers with MLP	Data distributions with many rare classes encourages ICL, while common repeated classes encourage IWL. ICL and IWL can coexist when the data distribution has both of these properties, like a Zipf distribution.
Singh et al. (2023)	Omniglot	12 layers with MLPs	ICL can deteriorate during training. L2 regularization of MLP weights especially can mitigate ICL transience.
Nguyen & Reddy (2024)	Gaussian random vectors mapped to binary labels	One attention layer with a three layer MLP	IWL and ICL circuits compete, with IWL being slower to develop. ICL transience appears when attention weights are L2 regularized more than MLP weights.
Anand et al. (2025)	Synthetic language data for syntax probing constructed from sampled words from the Penn Treebank 3 dataset (Marcus et al., 1993)	MultiBert checkpoints (Sellam et al., 2022)	Study "structural ICL" where the model cannot use any of the semantic properties of the context tokens to make predictions. This type of ICL is also transient. A variant of active forgetting (Chen et al., 2023) which scrambles the model's token embeddings is used just at the beginning of training to mitigate the transience of ICL.
Park et al. (2025a)	Finite mixtures of Markov chains	Two layers with MLPs.	IWL and ICL strategies compete with each other. ICL transience is due to IWL achieving better loss on training distribution data in the long run. When the model is performing the IWL strategy, Markov chain transition matrices from the training data can be recovered from MLP weights.
Singh et al. (2025)	Omniglot	Two layers, attention-only	shows that "context-constrained IWL" is asymptotically preferred and in their toy setting of Omniglot classification the ICL and CIWL mechanisms shared subcircuits. They found that ICL forms faster, but is not as favorable for the model to lower training loss asymptotically while the CIWL circuit takes a while to form. Furthermore, the embryonic development of the CIWL circuit allows for ICL to emerge. They used a trick specific to the Omniglot dataset, to fix the in-context exemplar to be the same as the query exemplar and showed that this made ICL asymptotically preferred.
Ku et al. (2025)	Sinusoidal regression and Omniglot	Four layers with MLPs	ICL and IWL transience is task specific and depends on the relative computational costs of each strategy. For example the Omniglot classification task requires high memorization capacity but is simple for ICL, therefore IWL is slow to develop. The opposite is true for the sinusoid regression task.
Chan et al. (2025)	Random vector inputs mapped to random label	Two layers with MLPs	Through the perspective of expected generalization error, ICL and IWL compete for which lowers the loss more. IWL is encouraged by data the repeats frequently and ICL is encouraged by rare data that can be reasoned about through context. ICL transience is explained by the rare data showing up enough times for it finally to be memorized.

Methods for controlling the development of different language model capabilities One practical way for modulating the capabilities of deep learning models is through model merging, where the weights of multiple models are combined in certain ways to produce a single model (Yang et al., 2024). Examples include, Maiti et al. (2025) using benchmark performance of individual models to determine expert models in certain domains and computes weighted averages of their weights to achieve a high performing and robust single model, Dekoninck et al. (2024) using model merging to control the style of text generation in language models, and Ro et al. (2023) showing that training an ensemble of vision models on portions of a dataset with a variety of difficulty levels and then merging, was a more sample efficient way to obtain a single model than training on the whole dataset.

A way of suppressing a certain capability from being learned during training is suppressing tokens in the dataset that correspond to that capability (Rathi & Radford, 2026). For post-training techniques, some are using "self-distillation" to mitigate catastrophic forgetting in models by leveraging their ICL capabilities (Shenfeld et al., 2026; Hübotter et al., 2026). To enhance a certain certain ability when finetuning a model Snell et al. (2022) proposes to distill context. Noticing that models perform better when given instructions in context and a scratchpad for generation before predicting the final output, context distillation consists of providing a model additional context and a scratchpad along with the training examples, then recording the outputs of the model. These outputs are then used as labels for finetuning. They show that this method makes models better able to internalize the instructions than just supervised finetuning.

Context-enhanced learning Context-enhanced learning corresponds to using the ICL capabilities of models to modulate their training. Similar to Snell et al. (2022) for finetuning, Qi et al. (2025) provides the training examples along with GPT-4 generated context that contains full information of the target to models, then collects their outputs to compute a "in-context editing loss" between the output conditioned on the additional context and the unconditional output. The model is then finetuned on the normal dataset, but this "in-context editing loss" is a regularization term in the training objective. They find that this process mitigates overfitting and leads to more natural outputs. Zhu et al. (2025) provides partial completions or intermediate steps towards the target of a training example to a model during finetuning and shows that models can get to better performance with exponentially less training examples. They also show that the additional context does not get memorized by the models. Finally, Lampinen et al. (2025) shows that ICL usually generalizes better than finetuning, so they use a model's ICL capabilities to augment finetuning datasets that lead to better generalization after a finetune.

B Extended Setup

Consider predicting the continuous-state of an *unknown* linear dynamical system. We focus[3] on the orthogonally evolved system family (Sander et al., 2024), where the system is defined by $U \in \mathbb{R}^{5\times5}$, a random orthogonal matrix. Each U is generated by the algorithm presented in (Mezzadri, 2006), which ensures a uniform sampling over all $\mathbb{R}^{5\times5}$ orthogonal matrices. The initial state is $\mathbf{x}_0 \sim \mathcal{N}\left(0, \frac{1}{5}I\right)$, with state updates:

$$\mathbf{x}_{i+1} = U\mathbf{x}_i = U^{i+1}\mathbf{x}_0. \tag{3}$$

The system state is in-principle perfectly predictable, but only after six positions in the sequence are observed by solving for

$$U = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5\end{bmatrix}\begin{bmatrix}\mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4\end{bmatrix}^{-1}. \tag{4}$$

[3]In the Appendix, we provide results for the identity system family which has dynamics that are even simpler than the orthogonal system family. For the identity systems, the initial state is $\mathbf{x}_0 \sim \mathcal{N}\left(0, \frac{1}{5}I\right) \in \mathbb{R}^5$. Now, the state updates as

$$\mathbf{x}_{i+1} = \mathbf{x}_i. \tag{2}$$

This trivial process of copying a constant is perfectly predictable after one realization is observed.

B.1 Optimal Pseudoinverse Predictor

Following from (4), given the state observations $\{\mathbf{x}_0, \ldots, \mathbf{x}_i\}$, an optimal predictor for this problem computes $\widehat{\mathbf{x}}_{i+1} = \widehat{U}\mathbf{x}_i$, where

$$\widehat{U} = [\mathbf{x}_1 \quad \ldots \quad \mathbf{x}_i] [\mathbf{x}_0 \quad \ldots \quad \mathbf{x}_{i-1}]^\dagger, \tag{5}$$

and $X^\dagger$ denotes the Moore-Penrose pseudoinverse of X. Essentially, this baseline only makes non-zero errors on the first, second, third, fourth, fifth, and sixth entry in any sequence — it gets everything else perfectly correct.

B.2 Data generation and training

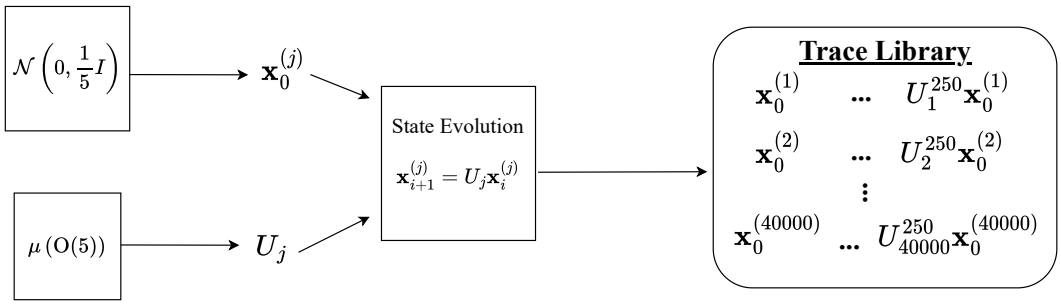


Figure 4: The generation of a train or test library of sequences.

Generating a library of training sequences: Depicted visually in Fig. 4, we first compile a training library by the following method:

1. Generate 40000 orthogonal matrices i.i.d. uniformly over all orthogonal matrices $U_1, \ldots, U_{40000} \overset{iid}{\sim} \mu\,(\mathrm{O}(5))$, where $\mu\,(\mathrm{O}(5))$ is the Haar measure over orthogonal matrices in $\mathbb{R}^{5\times5}$. (Mezzadri, 2006)
2. Generate 40000 i.i.d. initial states that will correspond to each training system $\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(40000)} \overset{iid}{\sim} \mathcal{N}\left(0, \frac{1}{5}I\right)$.
3. Roll out the states to get observation sequences that are each 251 entries long, and compile the sequences as our training library.

Cutting and interleaving training sequences To form a training trace, we interleave segments of observation sequences from the library into a context window of length 251, by this process:

1. Insert the start symbol at index 0.
2. Sample the maximum number of systems in the trace N from a $\mathrm{Zipf}(1.5, 25)$ distribution depicted graphically[4] in Fig. 5a. This means that no more than 25 systems will ever appear in a training trace.
3. Choose N of the 40,000 systems in the training library uniformly at random without replacement.
4. Randomly assign to each of the N systems a pair of symbolic open and close labels for this training example.
5. Sample the number of cuts $C \sim \mathrm{Poisson}(2N)$ to be made in the trace. This means that there will be $C + 1$ trace segments in the trace.[5]

[4]The Zipf distribution was chosen for its ubiquity in nature (Bak, 1996) and natural language (Schutze & Manning, 1999), along with recent work pointing to its importance in modern neural networks (Michaud et al., 2023).

[5]On average, each training trace has $2N + 1$ segments to ensure that the trained model has seen ample interruptions and continuations of systems.

6. Place the C cuts uniformly at random with replacement within the context window.
7. For each segment created by the cuts, in order, uniformly at random choose one of the N systems with replacement.
8. At the cut at the beginning of the segment, the open label for this segment's system is inserted.[6]
9. For the system chosen, check if it has appeared in a previous segment of the trace. If not, insert the system's segment from the training trace library starting at index 0. If this system has appeared in a previous segment, insert the system's segment from the training trace library starting at the index that corresponds to the continuation of the previous segment for this system.
10. At one index before the next cut, insert the close label for this segment's system.

Note that within a single training example, segments of a particular system always start with the same open token and always end with its corresponding close token. These random assignments are redrawn at the beginning of the interleaving process for each training example; therefore, *the same system can have different symbolic open and close labels when it appears in different training examples*. See Fig. 1 for a diagram of an interleaved training example.

Given this randomized procedure for generating interleaved training examples, we can analyze the training distribution to better understand how frequently a model must recall a system, or sees many systems in a trace.

In Fig. 5, we show relevant distributions that are derived from the randomized interleaving procedure in this section. Figs. 5a and 5c show the $\mathrm{Zipf}(1.5, 25)$ distribution for the maximum number of systems per trace in black and the number of unique systems per trace in silver. The frequency of number of unique systems per trace follows closely the $\mathrm{Zipf}(1.5, 25)$ distribution for the smaller quantities of systems, but diminishes quicker for larger numbers of systems, due to the coupon-collecting phenomenon of picking the same system multiple times in a trace. The PMF for the number of cuts made in an interleaved trace is shown in Fig. 5b, while the CCDF for this quantity is given in Fig. 5d. Fig. 5e shows the frequency of how many times a system appears in a training trace. If the same system appears more than once, than the model must perform recall. Therefore, Fig. 5e gives an idea of how often the model must recall a system during training. Finally, Fig. 5f provides the frequency of the number of previously seen systems in the training trace that are candidates to be continued when a predictor is tasked to recall a system. The value zero on the x-axis of this figure means that the model is seeing a system for the first time in a training example and has no need to recall. Later, in Section B.3, we construct tests for the associative recall ability of the trained model for different numbers of candidate systems to be continued in the trace. Fig. 5f shows that for 19 candidate systems, the largest number of candidate systems that we tested on, the model has been presented with this situation less than 1% of the time during training.

Input structure and embedding The input dimension of our models is 57. There are 50 dimensions for encoding paired symbolic open and close labels, a dimension for the start symbol, a dimension for the payload flag and 5 dimensions to hold the 5-dimensional observation vectors. The special symbols are one-hot encoded vectors; see Fig. 6 for an example of the open symbol. For the observation sequence between the SPLs, the 5-dimensional state vectors are inserted into the payload portion of the input vector, the payload flag is set to 1, and the rest of the input vector dimensions are zeroed out.

The entire randomized procedure of generating interleaved training traces is depicted in Fig. 7 along with the structure of the inputs into the model.

[6] Since the open and close labels occupy two indices in the context window, there are three special cases that can occur: (1) If two cuts are sampled to be on top of each other, then the first of the two cuts that were sampled is ignored; (2) If the two cuts are sampled to occupy adjacent indices, then only the close label for the system corresponding to first of the two cuts is inserted, effectively making that index meaningless as close labels are masked; (3) If the two cuts are sampled so that there is only one index between them, then the open label for the system corresponding to the first of the two cuts is inserted and is immediately followed by the close label for that system, effectively making both indices meaningless due to the masking of the labels. Note that the distributions shown in Fig. 5 do not account for these rare special cases.

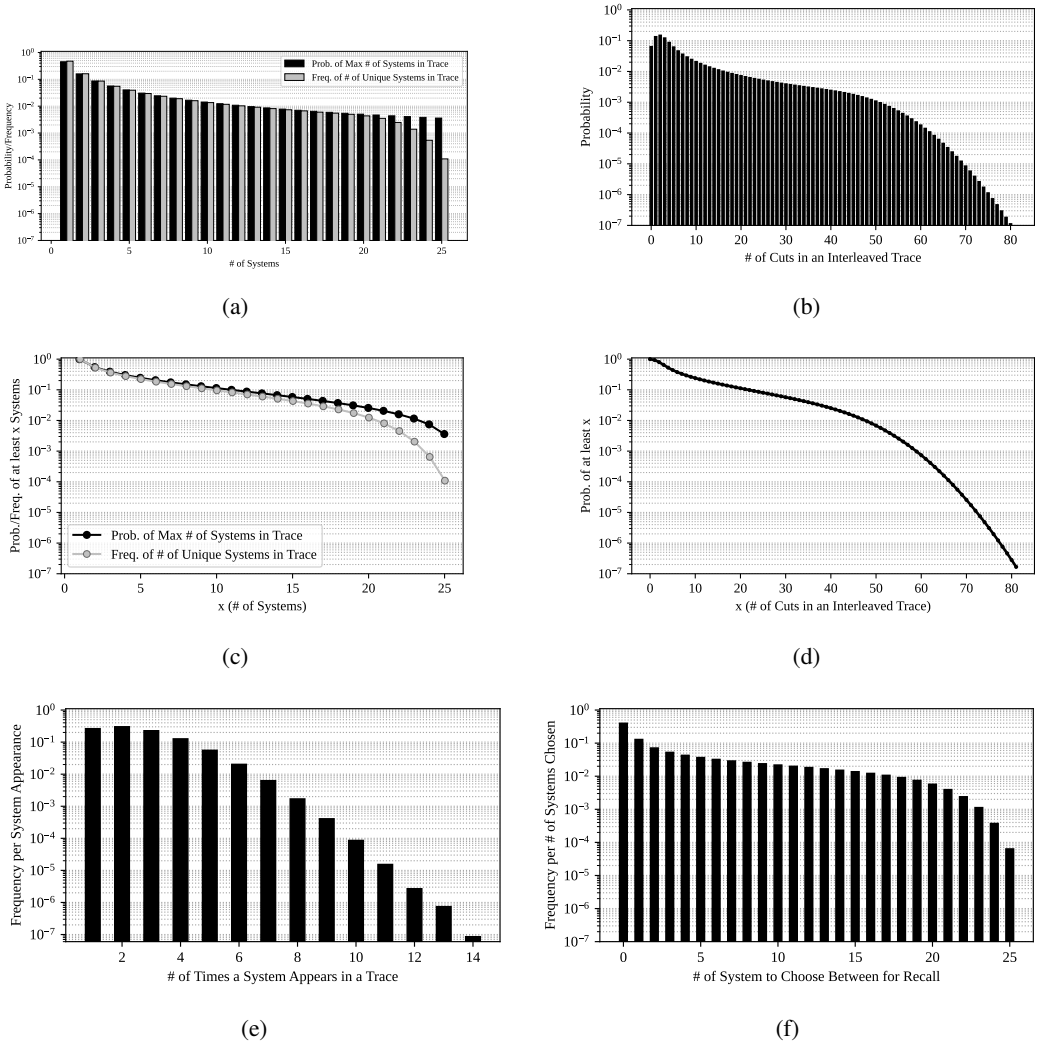


Figure 5: Distributions and complementary cumulative distribution functions (CCDFs) used in data generation — Fig. 5a is the $\text{Zipf}(1.5, 25)$ distribution for the maximum number of systems per trace in black and the number of unique systems per trace for 1×10^7 traces in silver. Fig. 5c shows the CCDFs for these distributions. Fig. 5b shows the PMF and Fig. 5d shows the CCDF of the number of cuts per trace. Fig. 5e shows the frequency of the number of times a system will appear in the same trace per system appearance. Lastly, Fig. 5f shows the frequency of the number of previously seen systems a predictor must choose between to recall in a trace per system appearance. For example, if system 0 is chosen then system 1, then system 0, then the model must choose between two systems to recall.

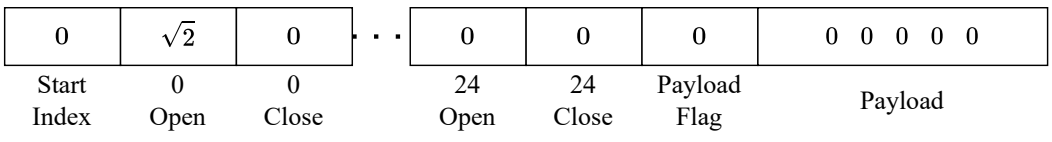


Figure 6: The one-hot encoding of an open symbolic label. In this example, the system corresponding to this label is assigned to be "system 0."

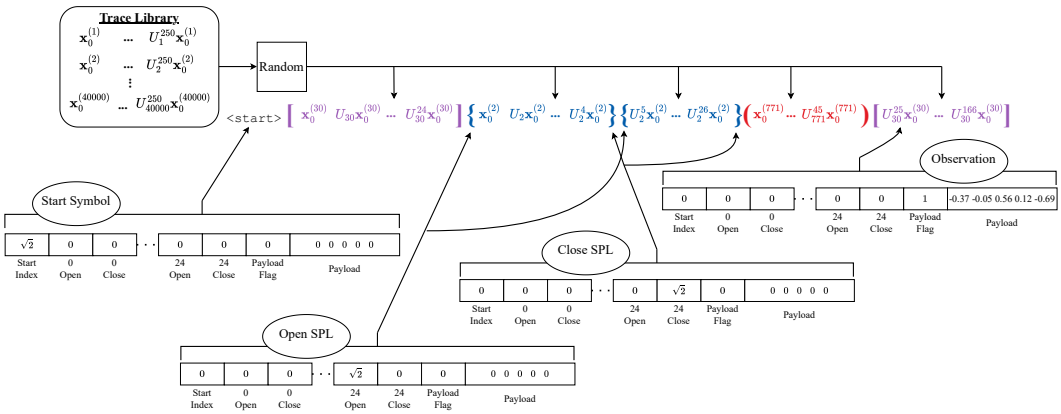


Figure 7: Generating a training example — Notice in this example the continuation from the first segment to the last (system U_{30}), and from the 2nd segment to the 3rd (system U_2). The "parentheses" (symbolic punctuating labels) are encoded as special tokens as shown.

Model and embedding Building off of the codebase in (Du et al., 2023), which was influenced by (Garg et al., 2022), we train a 10.7M parameter GPT-2 style transformer to perform this task. Our model has hidden dimension 192, 24 layers, and 12 heads. Our model's input embedding is 192×57 dimensional. The model's output layer is 5×192 to ensure that the model makes 5-dimensional predictions. The input and output layers are untied (Du et al., 2023). Three other smaller models were also trained, but we focus on the largest model because it exhibits in-weights learning much earlier in training than the others.

Training and Hyperparameters New interleaved training examples are generated for each training iteration and our GPT-2-style model was trained for next token prediction on these training traces.[7] The loss for all SPLs on the output were zeroed out. A model that successfully recalls the state of a system seen previously in its context should make predictions after the open token that perform as it would have if the relevant sequence had continued on without interruption.

Following the choice made in (Du et al., 2023), we trained our model with a weight decay of 1×10^{-2}. We used a batch size of 512, a learning rate of $\approx 1.58 \times 10^{-5}$, and trained on a single NVIDIA L40S GPU with 48GB of RAM. A single training run takes around 5 days. We used the AdamW Optimizer (Loshchilov & Hutter, 2019) and trained using mean-squared error loss.

B.3 Needle-in-a-haystack Test Setup

We explain our basic test setup to analyze the learning capabilities of models on the interleaved systems.

We generate 100 held-out systems and 1000 different held-out initial states[8] by the same method described in Section B.2 to form our testing library.

To evaluate the model's ability to recall a previously seen system, we generate a series of structured "needle-in-a-haystack" test traces through interleaving the traces in the testing library. A single "needle-in-a-haystack" trace is generated by the following procedure:

1. Choose $N \in [1, 19]$ to be the number of distinct systems in a test trace.
2. For each of the N systems, insert a segment of 10 state observations starting from index 0 from the testing library into the "needle-in-a-haystack" test trace. Each of these segments

[7] The model sees newly interleaved training examples at each iteration, but the training traces that are interleaved into the training example are drawn from the fixed training library of 40,000 sequences. Therefore, the model undergoes single-epoch training where the information within a training example might have appeared in many other training examples.

[8] We generate 1000 initial states for each system to narrow down the quartiles in the squared-error curves.

are individually punctuated with a unique open and close symbol pair. We call this portion of the trace the "haystack".

3. Append a query open symbol to the test trace that signifies which system in the haystack will be be continued. The segment that will be continued is called the "needle".
4. Append 10 state observations from the continuation of the system in the haystack corresponding to the query open symbol. This portion is called the "test segment".

See Fig. 8 for a diagram of a test trace for $N = 2$ systems in the haystack and system U_1 as the needle.

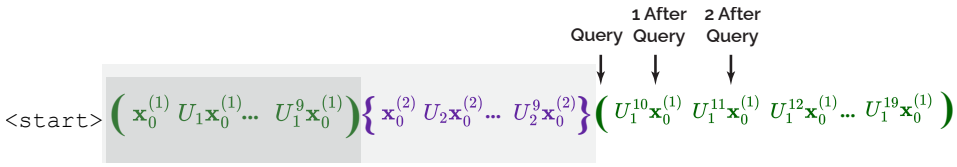


Figure 8: Needle-in-a-haystack test example. This shows a two system haystack, where the dark gray region corresponds to the needle segment and the light gray region corresponds to the entire haystack.

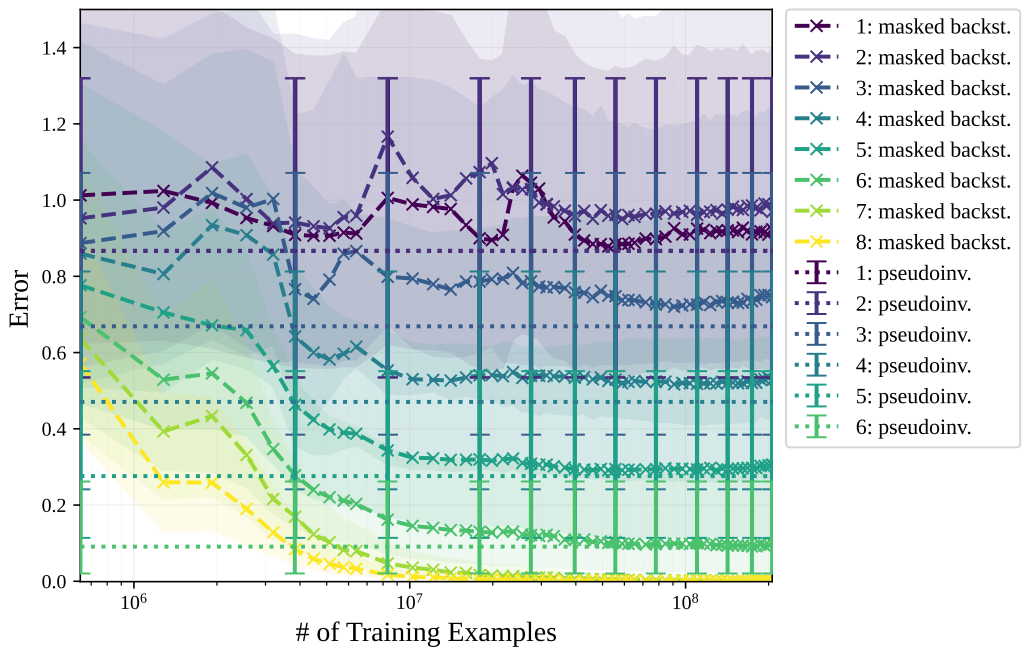
For the full "needle-in-a-haystack" test dataset, we would like to ensure that we test on the same systems for the different values of N, while having diverse systems in our dataset so that our results are statistically meaningful. To achieve this, we test on 50 "needle-in-a-haystack" trace configurations. A trace configuration is a specific ordering of systems from the testing library in the positions of the haystack. For the first needle-in-a-haystack trace configuration that we generate, we place a segment from "system 0" from the testing library into the first position in the haystack, and fill the rest of the haystack positions consecutively until the N^{th} position is filled with a segment from "system $N-1$". For the next trace configuration, the first position in the haystack is filled with a segment from "system 1" and the rest of the haystack positions are filled consecutively until the N^{th} position is filled with a segment from "system N". This pattern continues until the last trace configuration. In our case, we tested on 50 trace configurations, meaning the haystack of the last trace configuration started with "system 49" and ended with "system $48+N$". Each trace configuration is populated with 1000 different initial states for each system. For the results in the main paper, the test segment is a continuation of the segment in the first position of the haystack.

$$\begin{matrix} \{0 & \dots & N-1\} \\ \{1 & \dots & N\} \\ \vdots & \vdots & \vdots \\ \{49 & \dots & 48+N\} \end{matrix}$$

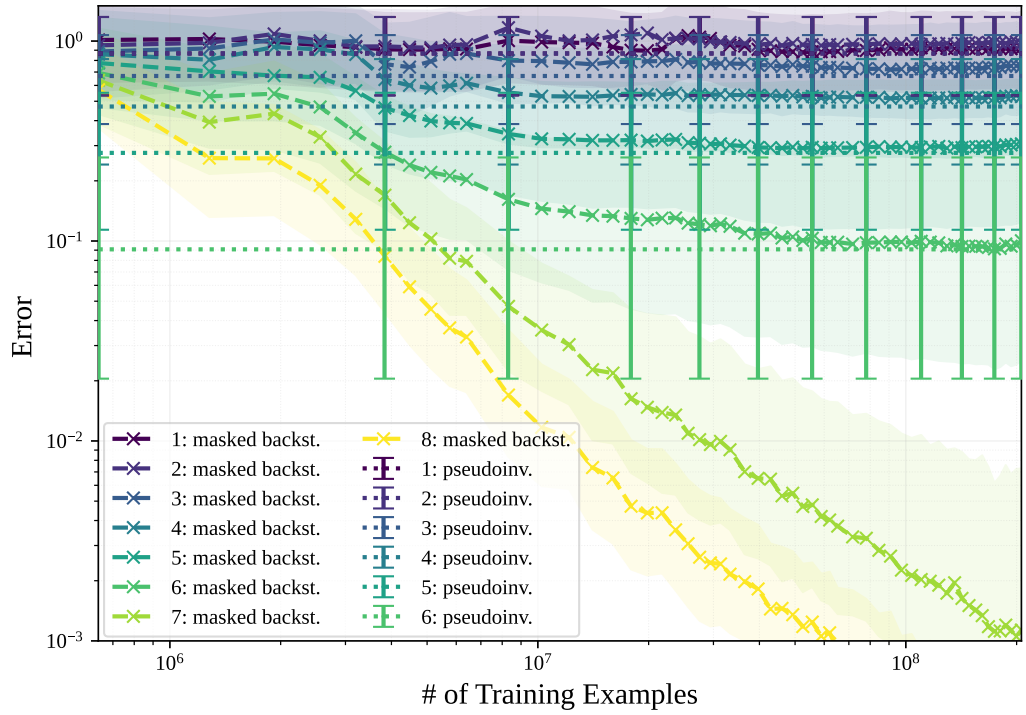
Figure 9: When testing on 50 needle-in-a-haystack trace configurations, the order of system indices from the testing library in a haystack of size N for each needle-in-a-haystack test trace is given above. For each system, 1000 sequences are interleaved to build a testing dataset of shape $50 \times 1000 \times (12N+1) \times 5$. The shape of the second to last axis is due to the start token and haystack segments being 10 context indices long plus two indices for the symbolic open and close labels. The last axis is 5-dimensional since every system has 5-dimensional observations.

C Training with Masked Backstories from the Beginning

Section 3.1 showed that when restarting training with masked backstories from a checkpoint that was trained using the normal interleaving procedure before the model's generalization performance deteriorates, the model's transition to in-weights learning is blocked, and the model's in-context associative recall abilities are improved. Now, we ask the question: what happens when we exclusively train a model using the masked backstories method?

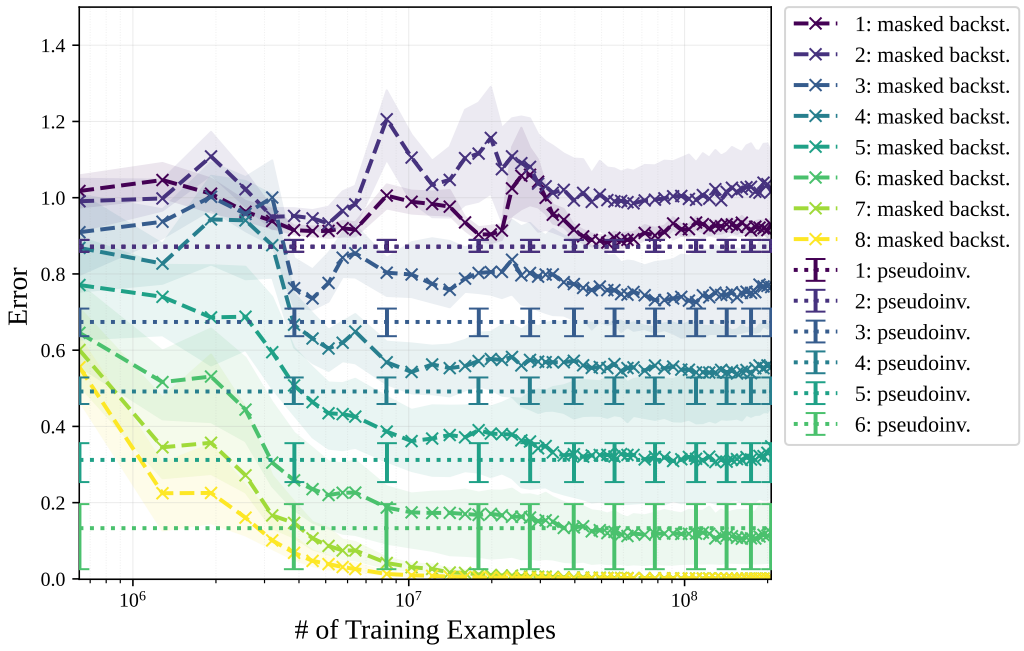


(a) Linear scale.

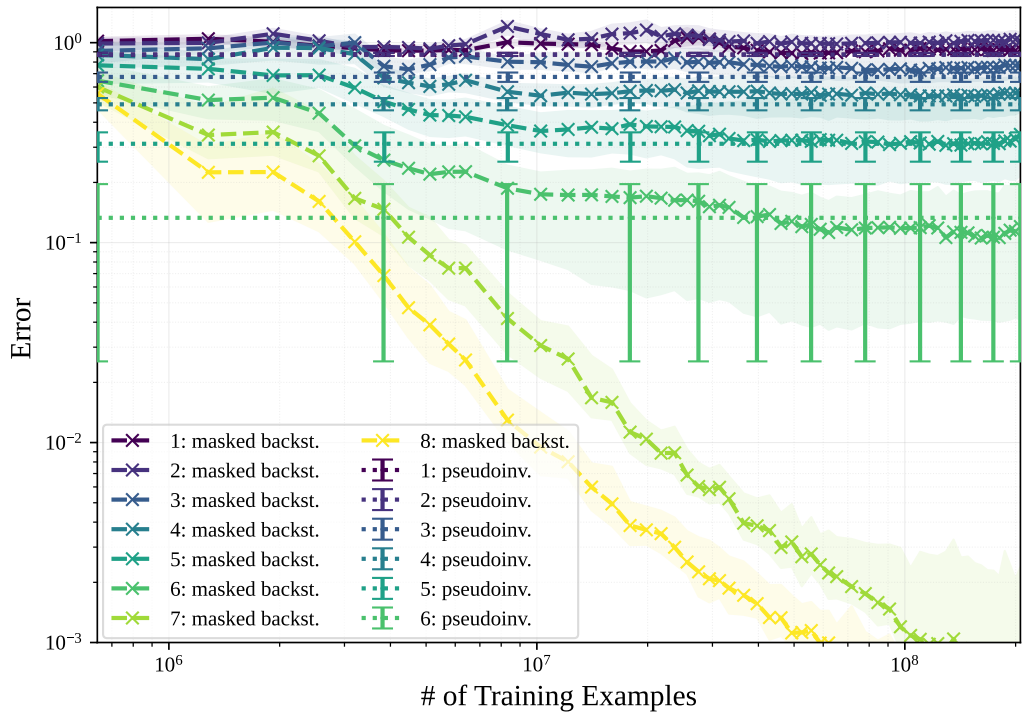


(b) Log scale.

Figure 10: Median squared error of the masked backstory trained model (crossed curves) and the pseudoinverse predictor (dotted lines) tested on backstoried training traces vs the number of training examples seen so far during training. The model's prediction error converges to the pseudoinverse predictor's error without any sudden shifts to in-weights learning.



(a) Linear scale.



(b) Log scale.

Figure 11: Median squared error of the masked backstory trained model (crossed curves) and the pseudoinverse predictor (dotted lines) tested on held-out traces vs the number of training examples seen so far during training. There is no degradation in the model's generalization performance.

First, we check if the model trained using masked backstories from the very beginning exhibits a transition to in-weights learning like the normally trained model. We check this by testing this model on the training traces with prepended backstories, since it is the prepended backstories now that have irreducible loss for this model. In Fig. 10, we plot the median squared error of the pseudoinverse predictor (dotted lines) and the transformer model (dashed curves with cross markers) for the first through the eighth index into the backstoried training traces. In this figure, we see that through 2×10^8 training examples seen during training the transformer model's median squared error converges down almost to the pseudoinverse predictor's median squared error without any sudden drop down to zero median squared error as we saw with the normally trained model at around 2×10^7 training examples in Fig. 13. Additionally, we also test the model on held-out "needle-in-a-haystack" traces and Fig. 11 shows the corresponding plot for generalization performance. Notice that the crossed curves remain horizontal and do not sharply increase through 2×10^8 training examples seen unlike the crossed curves in Fig. 14 at 3×10^7 training examples seen for the normally trained model. These two results together show us that the model trained with masked backstories model from the beginning does not transition to in-weights learning for the early indices up to this point in training.

Since this model was trained with masked backstories from the beginning, it was never penalized during training for poor predictions on the first 7 indices of an initial segment. Nonetheless, the model's median prediction error approaching the pseudoinverse predictor's median prediction error in Fig. 11 shows that this model still developed the ability to make predictions for these indices. Furthermore, one could think of the task of predicting the next vector in the segment when less than 6 observations have been seen as a more difficult task than predicting the next vector when there are enough observations to know the underlying system perfectly. This is because when there are not enough observations, the model must learn how to deal with the extra degrees of freedom caused by the underdetermined system of equations. Therefore, in this toy setting, the model is exhibiting a form of "easy-to-hard" generalization (Hase et al., 2024; Sun et al., 2024; Schwarzschild et al., 2021; Song et al., 2025), where its ability to predict the next vector perfectly when there are enough observations also allows it to reasonably predict the next token when it faces uncertainty.

The solid curves in Fig. 12 show the recall performance of the normally trained model (dot markers), the masked backstory trained model (diamond markers), and the unmasked backstory trained model (triangle markers). First notice that the diamond marked curves stay below the dot marked curves, which means the masked backstory trained model's associative recall ability is superior to the normally trained model's. Next looking at the triangle marked curves, in the beginning of training they follow improved performance of the masked backstory trained model's, but later in training around 3×10^7 training examples seen, the unmasked backstory trained model's improves at a slower rate than the masked backstory trained model. Training a model from scratch with masked backstories seems to lead to a decisive advantage in associative recall performance.

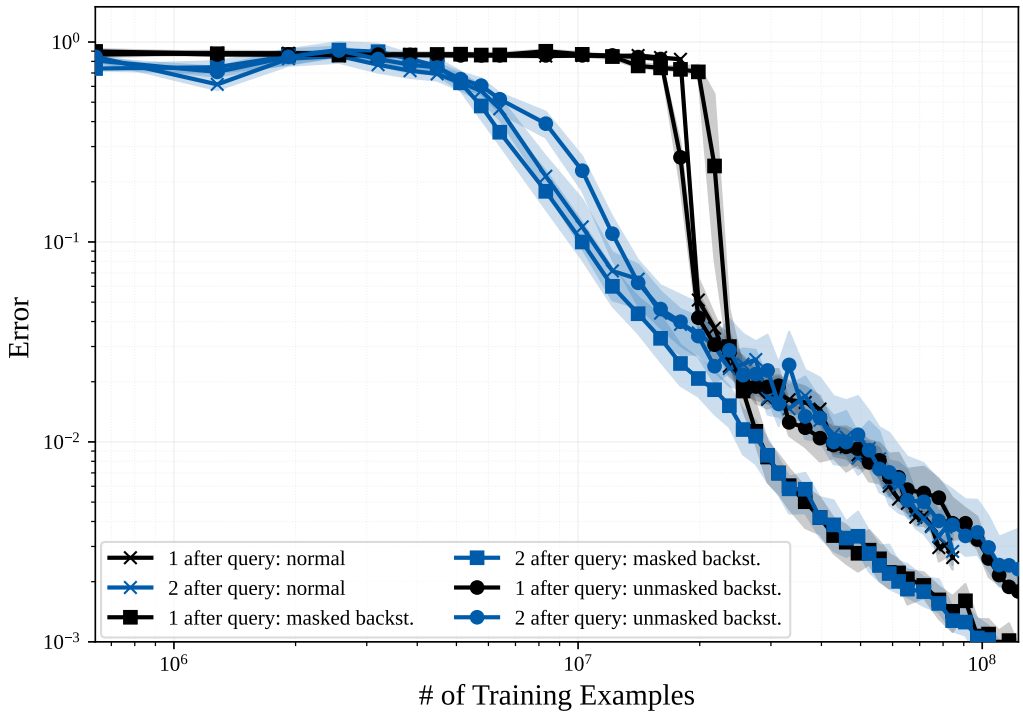


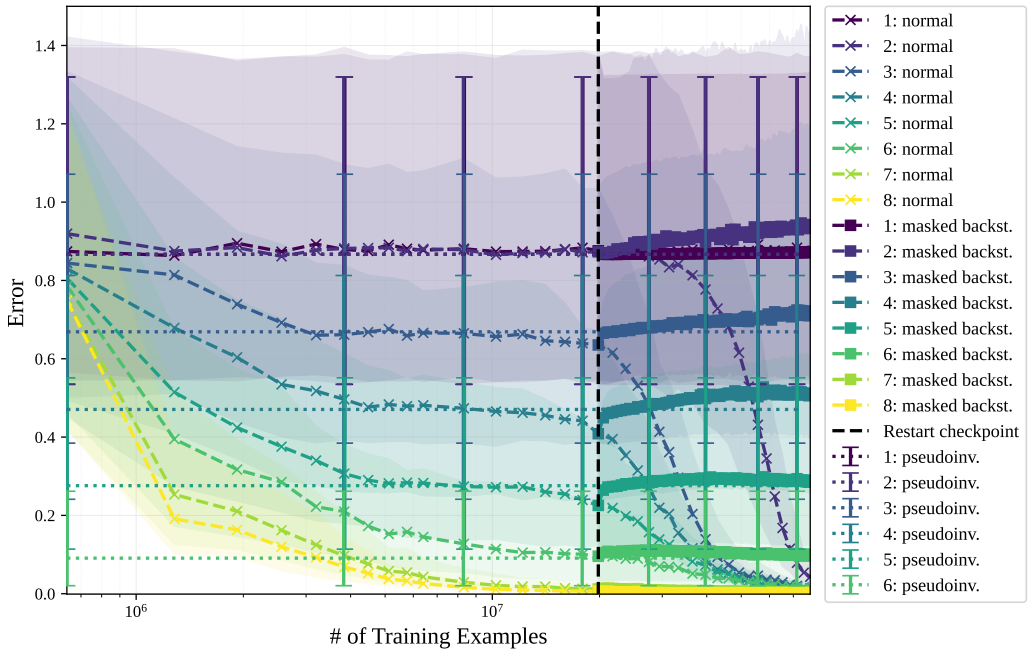
Figure 12: Median squared error of the normally trained model (dotted curves), the model trained with masked backstories from the beginning (diamond curves), and the model trained with unmasked backstories from the beginning (triangle curves) on indices one and two into the test segment of held-out "needle-in-a-haystack" traces with 5 systems in the haystack vs the number of training examples seen so far during training. Again, the masked backstory trained model's squared error (diamond curves) is always the lowest for the each index.

D More Figures for Restarting Training with Masked Backstories

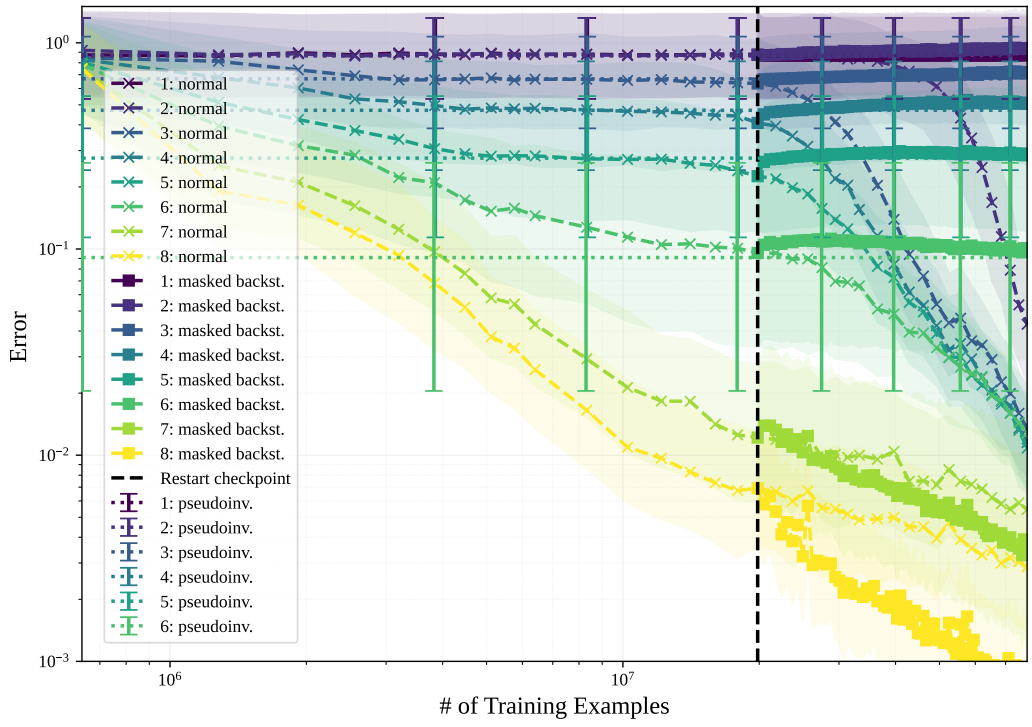
D.1 Teasing apart the effect of masking vs segment lengths

Further analyzing this positive effect on associative recall performance, we notice that augmenting the training traces with masked backstories means that the minimum length of a segment from a system that was previously unseen in the training example is 7, while the minimum length for the original interleaving procedure is zero (see Section B.2). A minimum length of 7 means that the full system is learnable by (4), and consequently, when queried for recall the model now more frequently recalls systems that it has more context about. To check if the improved performance is just due to this shift in the distribution of segment lengths in the training data, we trained a model with *unmasked* backstories. This model sees the same training data as the masked backstory trained model, the only difference is that the training loss on the backstory indices is not zeroed out. The performance of the unmasked backstory model shows us any ICL performance gains that can be made just by the shift in the segment length distribution.

In Fig. 15, the unmasked backstory model's recall performance is shown by the solid curves with triangle markers. Looking at the solid black curves for the first index into the test segment, the unmasked backstory trained model performs better than the normally trained model, but worse than the masked backstory trained model. Looking at the solid blue curves for the second index into the test segment, the unmasked backstory trained model performs at the same quality as the normally trained model, and worse than the masked backstory trained model. This shows that the improvements that we see from the masked backstory trained model cannot be solely attributed to the shift in the segment length distribution.

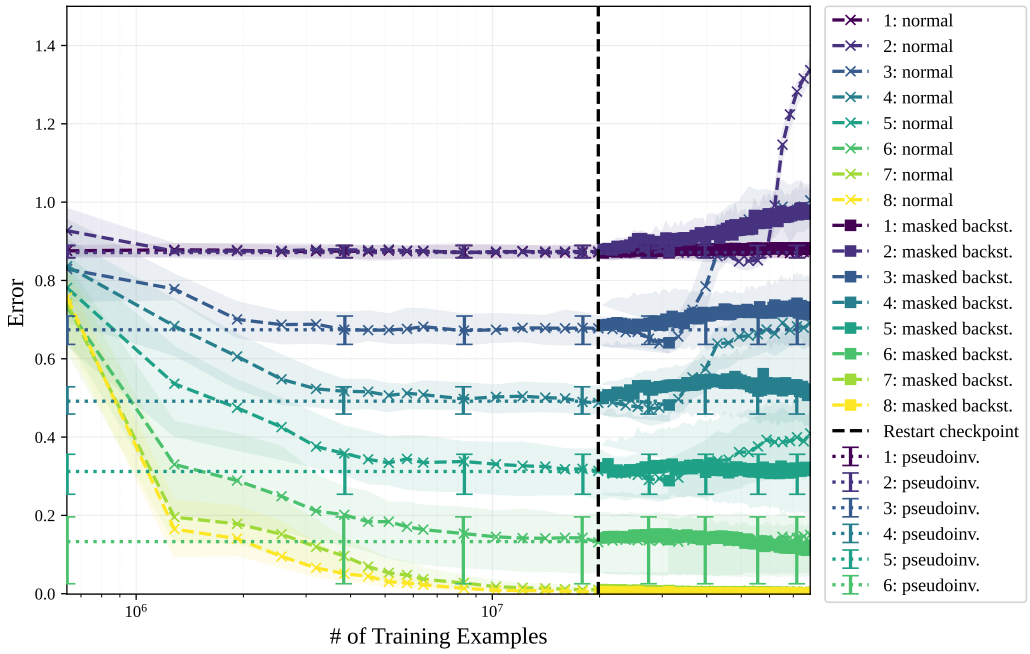


(a) Linear scale.

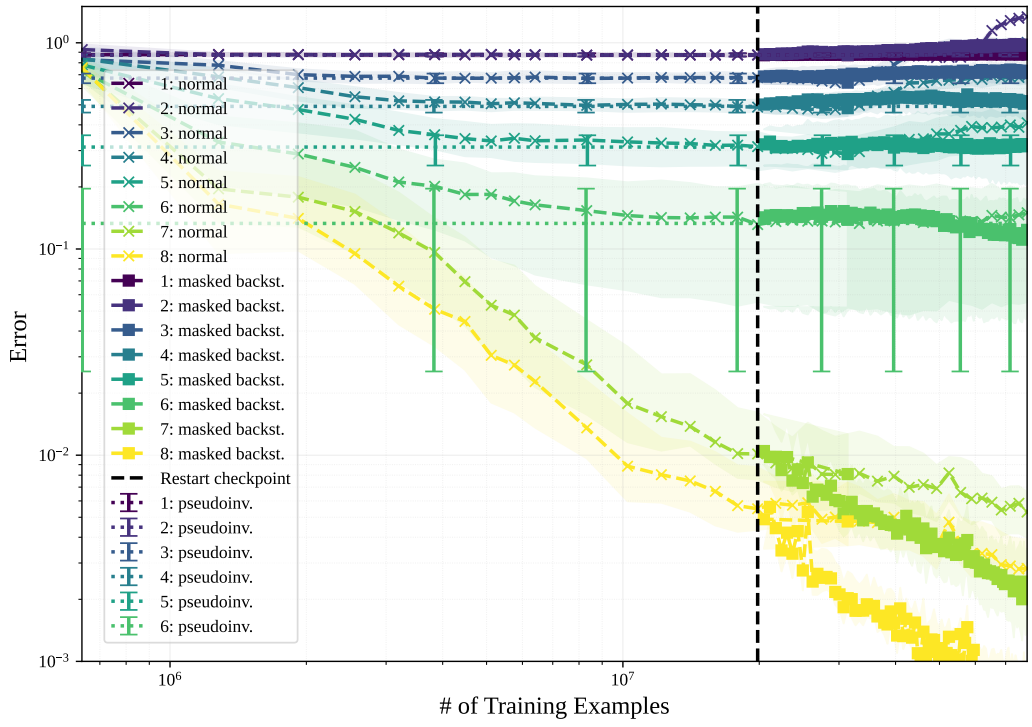


(b) Log scale.

Figure 13: Median squared error of the normally trained model (crossed curves), the masked backstory trained model (squared curves), and the pseudoinverse predictor (dotted lines) on specific indices into training traces vs the number of training examples seen so far during training. The crossed curves eventually sharply decrease towards zero (signifying a transition to in-weights learning) while the squared curves do not.

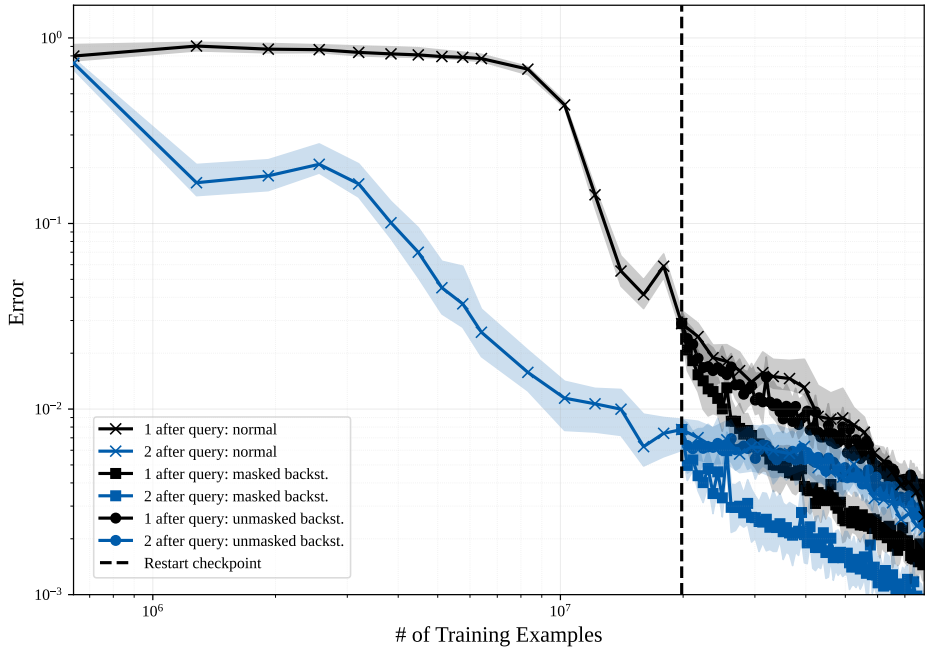


(a) Linear scale.

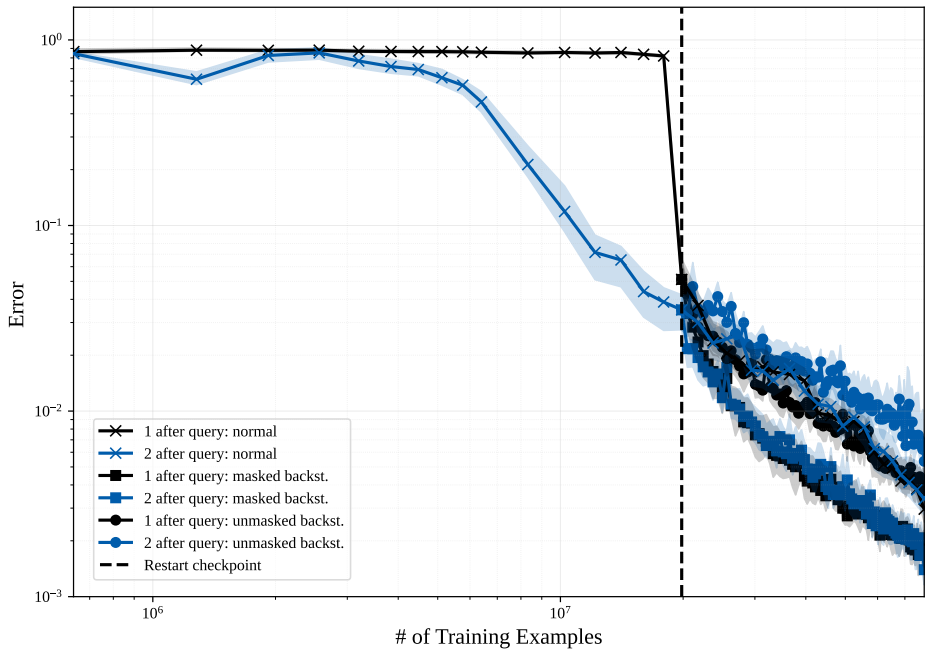


(b) Log scale.

Figure 14: Median squared error of the normally trained model (crossed curves), the masked backstory trained model (squared curves), and the pseudoinverse predictor (dotted lines) on specific indices into held-out traces vs the number of training examples seen so far during training. Notice that the crossed curves sharply increase late in training while the squared curves remain closer to their corresponding dotted lines.



(a) 1 system haystack.



(b) 5 system haystack.

Figure 15: Median squared error of the normally trained model (dotted curves), the masked backstory trained model (diamond curves), and the unmasked backstory trained model (triangle curves) on indices one and two into the test segment of held-out "needle-in-a-haystack" traces vs the number of training examples seen so far during training. See that the masked backstory trained model's squared error (diamond curves) is always the lowest for the each index.