Why Larger Models Learn More: Effects of Capacity, Interference, and Rare-Task Retention

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2026

Abstract

Larger models have been argued to learn tasks smaller models do not. What drives this phenomenon? We first develop a simple phenomenological argument that power-law scaling already suggests that a larger model will be able to learn a part of the data distribution that a smaller model fails to learn, even with infinite training data. To validate this claim and identify its causes, we study the effects of model scaling on a synthetic setup consisting of a mixture of tasks that show monotonic scaling curves. The results point to a data-induced competition over resources (neurons). Specifically, smaller models allocate their neurons to high frequency or low complexity tasks, and so they learn solutions that perform poorly on rare and complex tasks. Moreover, this happens even when solutions capable of expressing the desired task exist. We then assess how a larger model circumvents this data-centric bottleneck, finding that it traces to a reduced interference mechanism: larger models can allocate enough resources to common tasks that the gradient updates for those tasks become weak, which means that they do not overwrite rare-task features as they slowly accumulate. Finally, to further validate these claims, we pretrain OLMo models (4M to 4B parameters) on novel tasks of varying frequency and complexity. The results mirror those from our synthetic data experiments. Overall, we offer a data-centric account of why larger models learn tasks that smaller models fail to. This helps explain why larger models are better in practice, and it can inform practical questions concerning model sizing and training data mixtures.

1. Introduction

Prior work has claimed that the ability to solve certain critical tasks *only* emerges in larger models [2, 3, 11, 19, 30, 36, 38–40]. Such arguments have fueled the drive towards increased scaling. However, given the large training and inference costs that large models impose, it is worth identifying precisely what marginal benefits are unlocked by larger models and whether scaling parameters is the sole way of realizing those benefits. Our argument begins from the observation that **power-law scaling [17, 23, 35] already suggests that there is a regime in which a smaller model fails to learn parts of a data mixture that a larger model succeeds on, even under asymptotic training** (Sec. 2). Importantly, this is not an argument that larger models are simply more sample efficient [1, 11, 13, 15, 16, 18, 32, 34], but rather that smaller models suffer from a more fundamental limitation even under infinite training regimes. To validate this scaling-law prediction and identify its causes, we analyze a setting involving a mixture of regression tasks. In this, we are inspired by much recent work using toy tasks to pinpoint the effects of scaling [4–9, 12, 24, 26, 28, 29]. Furthermore, all of the individual tasks in our setting are learnable by the models under consideration. Correspondingly, mere expressivity notions are not the issue; instead, the question concerns the ability of these models to learn complex task distributions from data. These experiments lead to two key findings:

First, **scaling enables learning rare and complex tasks (Sec. 3.1).** We present an analytic argument that only larger models will (on average) learn the rare and complex tasks present in this setting. Second, **reduced competition for resources enables learning rare and complex tasks (Sec. 3.2)**. Only larger models, by virtue of having more parameters and hence less gradient interference, are able to retain memory of previously observed samples from a rare task. Thus, when the next batch of rare-task samples come in, the larger model builds on its prior knowledge, which ultimately leads to success despite the impoverished learning signal. Finally, we validate the above theoretical arguments in real LLMs (Sec. 4). Critically, the data-centric nature of our analysis suggests that understanding why larger models learn more requires not only asking what they can represent, but also what is learnable under gradient-based optimization from a given data mixture.

2. A Phenomenological Model Predicts Larger Models Learn More

Neural network scaling is known to predictably and monotonically improve loss: [18, 23, 33]:

$$L(N, D) = L_0 + \frac{A}{N^\alpha} + \frac{B}{D^\beta}, \tag{1}$$

where L_0 denotes the irreducible loss, A, B are constants, and α, β are parameter / data exponents ($\alpha \approx 0.46$ and $\beta \approx 0.51$ for Chinchilla-scaling [18]). Training in a compute-optimal manner, i.e., finding the model size and data configuration that helps achieve the minimum loss at a given compute budget C, gives us $L_{\mathrm{C}}(N) \propto N^{-\gamma}$, where $\gamma = 0.34$, and $L_{\mathrm{C}}(N)$ denotes the optimum loss achieved when training a model with N parameters under resource constraints. However, resource-constrained training by itself does not inform what a model can actually express. *Specifically, even though a smaller model may have a worse compute-optimal loss, we do not know if it is fundamentally incapable of achieving the same loss as the larger model.* To assess that statement, we must evaluate a model's loss under asymptotic resources (i.e., infinite data):[1] $L_\infty(N) \propto N^{-\alpha}$. We again see gains from merely scaling the model size: that is, *the asymptotic loss achieved by a larger model is better than the smaller one.* This indicates there is a part of the training distribution a smaller model, despite observing infinite data, fails to learn. This is the most interesting case that warrants further study: *What is it about the data that only a larger model can learn, such that the smaller model cannot, even after observing infinite data? How precisely does having more parameters aid this learning?*

3. Scaling Allows Learning Rare Tasks by Reducing Gradient Interference

Our phenomenological argument is based on monotonic (power-law) scaling—a phenomenon even synthetic tasks can recapitulate [4–9, 12, 24, 28, 29, 31, 33]. We thus follow this line of work and develop a multi-task learning setup that helps assess which tasks only a larger model can learn.

Data. We consider a multi-task learning setup where samples are drawn from a mixture of K linear regression tasks. Specifically, the k^{th} task is assumed to appear with *frequency* $\pi_k > 0$, such that $\sum_k \pi_k = 1$, and has *covariance* $C_k = B_k \Lambda_k B_k^\top = \sum_{j \geq 1} \lambda_{k,j}\, b_{k,j} b_{k,j}^\top$. Here, the "feature matrix" $B_k = [b_{k,1}, b_{k,2}, \ldots]$ is assumed to have orthonormal columns; $\Lambda_k = \mathrm{diag}(\lambda_{k,1}, \lambda_{k,2}, \ldots)$ with $\lambda_{k,1} \geq \lambda_{k,2} \geq \cdots \geq 0$; and different tasks occupy orthogonal blocks, i.e., $B_k^\top B_\ell = 0$ for $k \neq \ell$. Compared to prior work studying theory of scaling laws based on toy regression tasks, we emphasize that our setup involves the learning of multiple tasks simultaneously.

1. We note power-law scaling need not hold asymptotically [6, 31], which is why we call this argument phenomenological. It motivates the subsequent, rigorous claims.

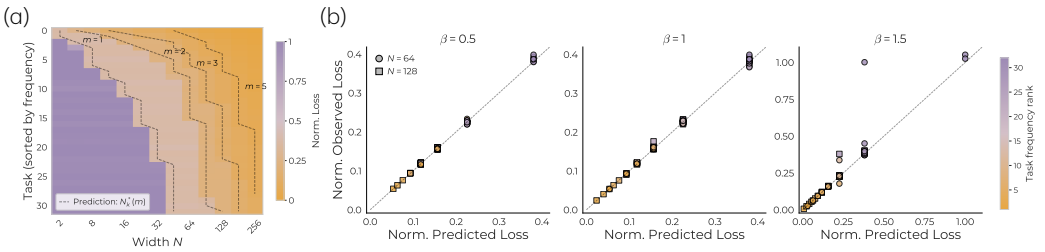

Figure 1: **Feature Utility Predicts Learning Order.** We train students of varying width on a mixture of $K = 32$ regression tasks with power-law task frequencies (β) and plot per-task loss (normalized by mean predictor). (a) Empirical phase diagram for which task features ($\beta = 1.0$) are retained as a function of width and task frequency match our prediction. (b) Loss matches the analytic prediction from Theorem 1 across task-frequency exponents. Overall, we see that increasing width preferentially improves low-frequency tasks because it allows the model to retain lower-utility features.

Teacher / Student Models. For a given input $x \sim \mathcal{N}(0, I)$, the teacher for task k is defined as $y_k = \Lambda_k^{1/2} B_k^\top x$. The student uses a shared width-N encoder $U \in \mathbb{R}^{d\times N}$, $U^\top U = I$, with projector $P_U = UU^\top$, together with task-specific linear decoders D_k to discern between tasks. The student prediction is $\hat{y}_k = D_k U^\top x$. The total mixture loss is the weighted sum $\mathcal{L}_N(U) = \sum_{k=1}^K \pi_k \ell_k(U)$, where $\ell_k(U) = \mathbb{E}\big[\|y_k - D_k U^\top x\|_2^2\big]$ is loss of the k^{th} task. We focus on the dynamics of the encoder.

3.1. Larger Models Learn Rarer, More Complex Tasks

In order to narrow down a mechanism that explains *how* larger models may be able to learn more, we must first identify precisely *what* it is that a larger model learns but a smaller one fails to. We begin with answering this question in our toy setup.

Theorem 1 (Features are Learned in Order of Utility) *For a given U, the mixture loss reduces to $L_N(U) = \mathrm{Tr}(M) - \mathrm{Tr}(U^\top M U)$, where $M := \sum_{k=1}^K \pi_k C_k$. Hence, a width-$N$ minimizer spans the top-N eigenspace of M, whose eigenvalues are defined by the weighted per-task spectra: $u_{k,j} := \pi_k \lambda_{k,j}$. Thus, the optimal encoder keeps the N features (k, j) with largest $u_{k,j}$—we call these terms utilities. This implies if $n_k(N)$ denotes the number of retained features from task k, then $\ell_k^*(N) = \sum_{j>n_k(N)} \lambda_{k,j}$. Conversely, the minimum width at which a model learns at least m features for all tasks is $N^*(m) = \min\big\{ N : n_k(N) \geq m \big\}$.*

This implies if a task is observed infrequently or it involves several features, e.g., if its spectrum decays very slowly, then (on average) only a larger model will learn it.

Verification. We verify the claim above by training our student model on a mixture of $K = 32$ tasks, using the Adam optimizer for 100K steps. Results are reported in Fig. 1 (also see App. D). We find (a) the per-task loss predictably reduces with model width, and (b) the residual loss for a given width matches the predicted value. Critically, we see larger models learning infrequent tasks better.

3.2. Scaling Reduces Interference and Allows for Retention of Rare Task Observations

We next analyze how width surmounts the challenge to learn low-frequency tasks. To this end, note that for the k^{th} task, the Riemannian gradient is $G_k(U) = 2(I - P_U)C_k U$, and hence the mixture gradient is $\dot{U} = 2(I - P_U)MU$. We then have the following claim.

Theorem 2 (Residual Controls Learning) *Let $\mathsf{F} \subseteq [K]$ denote the common or frequent tasks. Define these tasks' weighted covariance $M_\mathsf{F} := \sum_{k\in\mathsf{F}} \pi_k C_k$ and residual signal $\delta_\mathsf{F}(U) := \mathrm{Tr}\big((I - P_U)M_\mathsf{F}\big)$. Then, the aggregate common-task drift $G_\mathsf{F}(U) = 2(I - P_U)M_\mathsf{F}U$ obeys the bound $\|G_\mathsf{F}(U)\|_F \leq 2\sqrt{\lambda_1(M_\mathsf{F})\,\delta_\mathsf{F}(U)}$.*

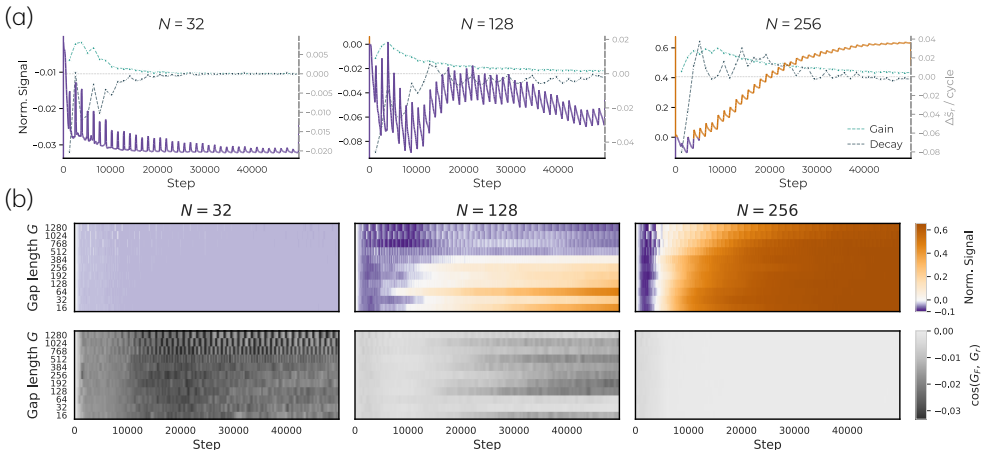

Figure 2: **Rare-Task Retention by Larger Models.** We isolate retention by training with a matched-frequency injection protocol: the rare task is withheld for G steps and then reintroduced in a batch such that its overall frequency is consistent across settings. (a) Training dynamics for $G = 1280$. We see small models briefly encode the rare task (Norm. signal $\tilde{s_r}$: left-y axis) after each injection; specifically, $\Delta\tilde{s_r}$ increases at point of injection, as shown by green dotted line ('gain'). However, as frequent-task updates resume, this signal is quickly lost ('decay': gray dotted line). Meanwhile, larger models retain more of the rare-task signal between injections and accumulate it over training. (b) Across injection gaps G and widths N, rare-task signal decays rapidly in narrow models but remains stable in wider models, while frequent-task signal is largely unaffected. Furthermore, by computing the cosine similarity of gradients via a batch of rare-task samples G_r and frequent task samples G_{F}, we see scaling provides enough representational capacity such that updates from frequent tasks no longer overwrite rare-task features before the next rare observation arrives.

The statement above says a set of tasks move the model only through the part of their covariance that is *not already explained* by the current representation, i.e., the residual $\delta_{\mathsf{F}}(U)$. *This leaves any spare width available to rare-tasks.* More precisely, let $\mu_1^{\mathsf{F}} \geq \mu_2^{\mathsf{F}} \geq \cdots$ be the eigenvalues of M_{F}. The best width-N representation for the common tasks alone leaves residual $\delta_{\mathsf{F}}^*(N) = \sum_{i>N} \mu_i^{\mathsf{F}}$.

Corollary 3 (Width-Scaling Reduces Interference) *Define* $N_{\mathsf{F}}(\varepsilon) := \min\left\{N : \sum_{i>N} \mu_i^{\mathsf{F}} \leq \varepsilon\right\}$. *For every* $N \geq N_{\mathsf{F}}(\varepsilon)$*, there exists an encoder for which* $\delta_{\mathsf{F}}^*(N) \leq \epsilon$ *and* $G_{\mathsf{F}}(U) \leq 2\sqrt{\mu_1^{\mathsf{F}}\epsilon}$.

Via Theorem 2, we get the following: Once $N \gtrsim N_{\mathsf{F}}(\varepsilon)$, the model contains enough resources that can be allocated to the common tasks, rendering the gradient towards them weak. However, even once interference is weak enough for a rare task to be learned, it is unclear whether gradient descent can actually consolidate that signal across its infrequent observations. We next characterize the local condition under which a specific rare feature can be learned, without forcing the forgetting of well-learned tasks. Specifically, assume we wanted to learn a rare rank-one task $C_r = \lambda_r b_r b_r^\top$ orthogonal to the common block. Let $U_F^{(N)}$ be top-N Eigenspace of M_F with eigenvalues $\mu_1^F \geq \mu_2^F \geq \cdots$.

Proposition 4 (Local Competition) *The common-task solution* $U_{\mathsf{F}}^{(N)}$ *is stable against direction* b_r*, i.e., common tasks' loss does not grow by learning of* b_r*, iff* $\pi_r\lambda_r < \mu_N^{\mathsf{F}}$*. Thus, the critical width at which* b_r *gets learned is* $N_r^{crit} := \min\{N : \mu_N^{\mathsf{F}} \leq \pi_r\lambda_r\}$.

This suggests the learning bottleneck is defined by the interaction between data and scale: *by increasing width, one avails capacity to such low-utility tasks and reduces competition between tasks*

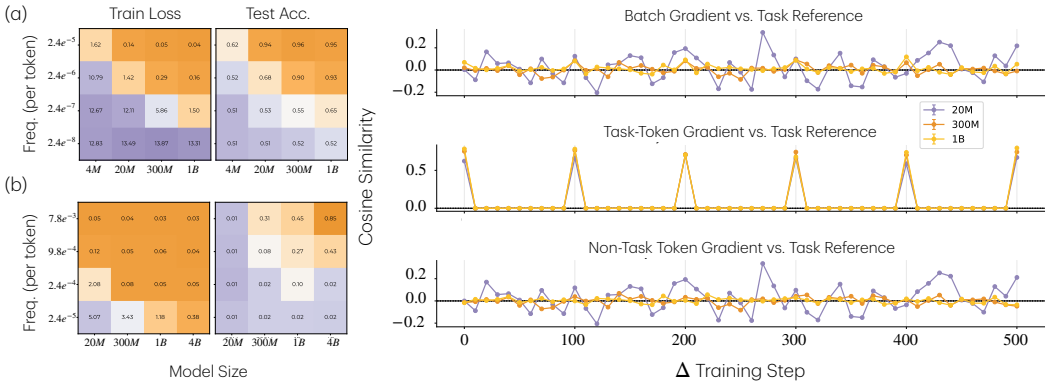

Figure 3: Left: **Larger Models Learn Rare Tasks; Smaller Models Do Not.** We visualize training loss and test accuracy for the two injected tasks. Right: **Gradient Interference.** The batch gradient of larger models carry more task signals with little to no interference. Overall, we see that increasing width enabling learning of low-frequency tasks by reducing gradient interference.

over model parameters, enabling learning of the rare task without forcing the forgetting of features relevant to common tasks.

Validation. To isolate how the gap between observations interacts with width, we design a matched-frequency injection experiment as shown in Fig. 2. This emphasizes the ability of a model to retain memories about observed data, while preserving the total frequency with which it is seen. We see at the at end of training, rare-task signal decays monotonically with G at all widths, but far more steeply for smaller models. Meanwhile, the learning dynamics in panel (b) show that after each injection, a larger model accumulates rare-task signal and retains enough of it to build on the next injection, while a smaller model decays back to near-zero in between.

4. Corroborating Claims with the OLMo Pretraining Pipeline

We now verify the claims of Sec. 3 in a realistic LLM pre-training setting using the OLMo pipeline. We train models of size 4M to 4B on up to 210B tokens (∼50K steps) by injecting instances of two novel tasks into pre-training data at controlled frequencies. Details can be found in App. A.2.
Fig. 3 validates two key claims in Sec. 3: (1) Larger models can learn rarer tasks that small models cannot learn, which replicates Fig. 1 (2) Larger models have less gradient interference between general language modeling task and the injected task, which is inline with Fig. 2.

5. Discussion

We develop a data-centric account of why larger models can learn tasks that smaller models fail to learn. Specifically, we show that larger models can learn rare tasks from the data mixture, and this phenomenon is explained by learning dynamics, i.e., competition of resources and retention of memories, as well as the task frequency and complexity. Our perspective highlights that understanding scaling requires thinking beyond model expressivity. We need to understand how learning dynamics are at play with task frequency and complexity. It also points toward more intentional design of data mixtures to better elicit target capabilities. For example, simply scaling up the frequency of a target task might provide a more efficient way to learn the task than scaling up the model size.

References

[1] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

[2] Anthropic. Responsible Scaling Policy, 2026. https://www.anthropic.com/responsible-scaling-policy.

[3] Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023.

[4] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

[5] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*, 2024.

[6] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. How feature learning can improve neural scaling laws. *Journal of Statistical Mechanics: Theory and Experiment*, 2025(8):084002, 2025.

[7] Francesco Cagnetta, Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Scaling laws and representation learning in simple hierarchical languages: Transformers vs. convolutional architectures. *arXiv preprint arXiv:2505.07070*, 2025.

[8] Francesco Cagnetta, Hyunmo Kang, and Matthieu Wyart. Learning curves theory for hierarchically compositional data with power-law distributed features. *arXiv preprint arXiv:2505.07067*, 2025.

[9] Francesco Cagnetta, Allan Raventós, Surya Ganguli, and Matthieu Wyart. Deriving neural scaling laws from the statistics of natural language. *arXiv preprint arXiv:2602.07488*, 2026.

[10] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-06-9. URL https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

[11] Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. *arXiv preprint arXiv:2403.15796*, 2024.

[12] Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in deep feature learning: Data, compute, width, and luck. *Advances in Neural Information Processing Systems*, 36:48021–48034, 2023.

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024.

[15] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[16] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

[17] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[18] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[19] Shengding Hu, Xin Liu, Xu Han, Xinrong Zhang, Chaoqun He, Weilin Zhao, Yankai Lin, Ning Ding, Zebin Ou, Guoyang Zeng, Zhiyuan Liu, and Maosong Sun. Unlock Predictable Scaling from Emergent Abilities, 2023.

[20] Jing Huang, Diyi Yang, and Christopher Potts. Demystifying verbatim memorization in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10711–10732, 2024.

[21] Hyeonbin Hwang and Yeachan Park. Intrinsic task symmetry drives generalization in algorithmic tasks, 2026. URL https://arxiv.org/abs/2603.01968.

[22] Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7bJizxLKrR.

[23] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[24] Licong Lin, Jingfeng Wu, Sham M Kakade, Peter L Bartlett, and Jason D Lee. Scaling laws in linear regression: Compute, parameters, and data. *Advances in Neural Information Processing Systems*, 37:60556–60606, 2024.

[25] Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=u2vAyMeLMm.

[26] Ekdeep Singh Lubana, Kyogo Kawaguchi, Robert P Dick, and Hidenori Tanaka. A percolation model of emergence: Analyzing transformers trained on a formal language. *arXiv preprint arXiv:2408.12578*, 2024.

[27] Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, Noah A. Smith, Pang Wei Koh, and Jesse Dodge. DataDecide: How to Predict Best Pretraining Data with Small Experiments. *arXiv preprint*, 2025.

[28] Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

[29] Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] OpenAI. Our Approach to Frontier Risk, 2023. https://openai.com/global-affairs/our-approach-to-frontier-risk/.

[31] Elliot Paquette, Courtney Paquette, Lechao Xiao, and Jeffrey Pennington. 4+3 phases of compute-optimal neural scaling laws. *Advances in Neural Information Processing Systems*, 37: 16459–16537, 2024.

[32] Tim Pearce and Jinyeop Song. Reconciling kaplan and chinchilla scaling laws. *arXiv preprint arXiv:2406.12907*, 2024.

[33] Shikai Qiu, Lechao Xiao, Andrew Gordon Wilson, Jeffrey Pennington, and Atish Agarwala. Scaling collapse reveals universal dynamics in compute-optimally trained neural networks. *arXiv preprint arXiv:2507.02119*, 2025.

[34] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[35] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

[36] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[37] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters,

Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024. URL https://huggingface.co/datasets/allenai/dolma.

[38] Jason Wei. 137 emergent abilities of large language models, 2022. https://www.jasonwei.net/blog/emergence. Accessed on: October 20, 2023.

[39] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[40] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

[41] Johnny Wei, Ameya Godbole, Mohammad Aflah Khan, Ryan Yixiang Wang, Xiaoyuan Zhu, James Flemings, Nitya Kashyap, Krishna P. Gummadi, Willie Neiswanger, and Robin Jia. Hubble: a model suite to advance the study of LLM memorization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZfdnZhOP0k.

Appendix A. Experimental Details

A.1. Synthetic Experiment

In the following, we describe experiment details and metrics relevant to results for the synthetic setup.

Data-generating process. All synthetic runs use the orthogonal-block instantiation of the mixture-of-regressions setup proposed in Sec. 3. We fix the ambient dimension at $D = 1024$, the number of tasks at $K \in \{16, 32\}$ (almost all figures use $K = 32$), and a per-task block dimension d_T such that $K \cdot d_T \leq D$ and the task blocks are mutually orthogonal. Concretely, task k occupies coordinates $[k\, d_T,\ (k{+}1)\, d_T)$ of $\mathbb{R}^D$, and its within-block spectrum is the power-law $\sigma_{k,j} = j^{-\alpha_k}$ for $j = 1, \ldots, d_T$. Unless stated otherwise we use a shared exponent $\alpha_k \equiv \alpha$ across tasks ($\alpha = 1$ in the orthogonal-block experiments, making the within-block decay slow enough that capacity reliably spreads beyond the leading mode of each task). The task prior is the power-law $\pi_k \propto k^{-\beta}$, normalized to sum to one over $k = 1, \ldots, K$; $\beta = 2$ in most experiments. Inputs are sampled fresh each step as $x \sim \mathcal{N}(0, \sigma_{\text{in}}^2 I_D)$ with $\sigma_{\text{in}} = 1$ in all orthogonal-block runs. The per-task targets are $y_k = \Lambda_k^{1/2} B_k^\top x$, restricted to the task's block; because each task block has rank d_T, the output dimension of the regressor is d_T (and reduces to 1 when $d_T = 1$, e.g., in the rank-1 specialization of App. D.1.2).

Model. The student is the linear-bottleneck regressor of Sec. 3: a shared encoder $W \in \mathbb{R}^{N \times D}$ that maps the input to an N-dimensional hidden, followed by per-task linear decoders $D_k \in \mathbb{R}^{d_T \times N}$ selected by the ground-truth task index supplied in the batch. We do not explicitly constrain W to have orthonormal rows: the relevant object for Theorem 1 is the projector $P_W = W^\top (W W^\top)^{-1} W$, which is invariant to the right-multiplicative gauge of W and which gradient flow drives toward the top-N eigenspace of $M = \sum_k \pi_k C_k$ regardless of the parametrization. The encoder is initialized such that $W^\top W = I_N$ at step zero, and the per-task decoders are initialized with Kaiming-uniform fan-in / linear gain. The decoders are jointly optimized with the encoder rather than analytically closed-formed at each step, since learned decoders will converge to $D_k^* = \Lambda_k^{1/2} B_k^\top U$ at any stationary point, so the joint optimization does not change the encoder fixed point but does match the practical setting in which both ends of the bottleneck are learned simultaneously.

Optimizer. We use AdamW with default hyperparameters and an inverse-square-root learning-rate schedule. Gradients are clipped at maximum norm 1.0. Batches are drawn fresh each step (no fixed dataset, no replay) with batch size $B = 1\,024$ for the phase-diagram and rank-1 sweeps, and $B = 512$ for the matched-frequency retention sweeps; the smaller batch in the retention runs is required so that an injection batch with $m \leq B$ rare-task slots can match the long-run frequency $\rho_r = m/(G \cdot B)$ at the $\rho_r \approx 6 \times 10^{-4}$ end of the sweep.

Metrics. We track three families of metrics, all reported on freshly sampled batches separate from the training stream. The first is the *per-task loss*, i.e., the unnormalized population MSE $\ell_k(U) = \mathbb{E}\big[\|y_k - D_k U^\top x\|_2^2\big]$, and its normalized counterpart $\ell_k(U)/\ell_{k,\text{baseline}}$ with $\ell_{k,\text{baseline}} = \|a_k\|_2^2/d_T$ the mean-predictor MSE per task. The second is the *per-task subspace alignment*, the basis-free quantity $s_k(U) = \text{Tr}(P_U C_k)/\text{Tr}(C_k) = \|P_U a_k\|_2^2/\|a_k\|_2^2$, computed via the SVD of W so that it is independent of the gauge of the encoder. s_k lies between N/D at random initialization and 1 when the task block is fully captured. We also report its random-baseline-corrected normalization $\tilde{s}_k(U) = (s_k(U) - N/D)/(1 - N/D)$, which equals 0 at random initialization and 1 at full capture. The third is the *residual common-task signal*: we compute $\delta_F(U) = \sum_{k \in F} \pi_k\, (1 - s_k(U))\, \|a_k\|_2^2$, the

residual energy of the frequent block. The frequent set F is the smallest top-prior set with cumulative mass at least 0.8; under our power-law prior this yields $|F| = \{6, 3, 2, 2\}$ for $\beta \in \{0.5, 1.0, 1.5, 2.0\}$ respectively. Standard evaluation is performed every $1\,000$–$2\,000$ steps on a held-out probe of the same population distribution, with the final checkpoint additionally re-evaluated for end-of-training summary statistics.

A.2. OLMo Pretraining Pipeline

Model Name	# Parameters	# Layers	Hidden Dim	MLP Dim	# Attn Heads
4M	6,963,200	8	64	512	8
20M	28,753,920	16	192	1,536	8
300M	371,458,048	16	1,024	8,192	16
1B	1,279,787,008	16	2,048	16,384	16
4B	4,707,057,664	16	4,096	32,768	32

Table 1: Model configurations by size.

Models. We use the OLMo model architecture [14]. For 4M to 1B models, we follow the model configuration and naming convention of Magnusson et al. [27]. We additionally include a 4B model to further evaluate width scaling.

Training hyperparameters. We use the same batch size of 1024, window size of 4096 for T_{CMP} and 1024 for T_{ADD}, and a learning rate schedule with an initial learning rate of 3×10^{-4} and cosine with warmup schedule for all models. For the retention window ablation experiment, we use a smaller window size of 512 to reduce the training cost. For a full list of hyperparameters, refer to the `OLMo-7B-0724` configuration.[2]

Training pipeline. We use the OLMo code base.[3] Our usage is inline with its Apache-2.0 license.[4]

Compute resources. All models are trained on a cluster of NVIDIA H200 GPUs.

A.3. Pre-training and Injected Task Data

A key variable in our claims is the frequency of a task. However, measuring the frequency of a natural occurring task in pre-training data is challenging, as instances from the same task can occur in many surface forms. To tightly control task frequency, we adopt a data injection framework from the memorization literature [10, 20, 22, 41]. We inject different instances sampled from the distribution of a special task T at a controlled frequency f to measure whether a model has learned the task distribution. The task T is special in the sense that it is unlikely to be part of normal pre-training data. We then train models of various size on data mixtures generated from different values of f.

Data. We use Dolma v1.7 as the pre-training corpus [37]. Given a task T, we inject instances sampled from its train split at a frequency of 7.8×10^{-3} to 2.4×10^{-8}, roughly from 1K instances per batch to 1 instance every 10 batches. To ensure the injected task frequency is comparable to the frequency of tasks learned in pre-training, we sample two reference tasks R_{cmp} and R_{add} from pre-training that involve similar high-level functions. The three-token sequence plus an end of document token replace the first four tokens of a training sequence. Details are in Appendix A.3

Pre-training data. We use Dolma v1.7 as the pre-training corpus [37]. Specifically, we use the 210B tokens corresponding to the first 50K batches that `OLMo-7B-0424` and `OLMo-7B-0724` are trained on, in the exact same order.

Reference Tasks. To ensure the injected task frequency is comparable to the frequency of tasks learned in pre-training, we sample two reference tasks R_{cmp} and R_{add} from pre-training that involve

2. https://github.com/allenai/OLMo/blob/main/configs/official-0724/OLMo-7B-0724.yaml
3. https://github.com/allenai/OLMo
4. https://github.com/allenai/OLMo?tab=Apache-2.0-1-ov-file

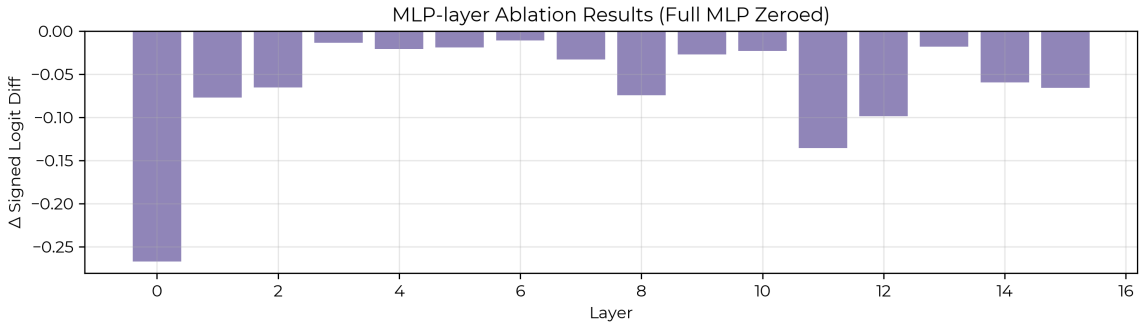

Figure 4: The first MLP layer has the strongest causal effects on the model's logits prediction.

similar high-level functions. R_{cmp} predicts a number larger than `x` in the prompt "it has increased from `{x}` to". R_{add} predicts the sum of two numbers smaller than 100 with the prompt "`{x} + {y} =`". We estimate the (lower bound of) their frequency in pre-training data using infini-gram [25] and observe models' next token prediction loss on the task.

Inject tasks. We consider two special tasks T: comparison (T_{CMP}) and modular addition (T_{ADD}). Both tasks are encoded as a sequence of three tokens: `TOK1`, `TOK2`, `LABEL`, where `TOK1`, `TOK2` $\in \mathcal{S}$, a set of 100 tokens randomly sampled from the vocab. Let $val(\cdot) : \mathcal{S} \mapsto [0, 99]$ be a bijective mapping that assigns an integer value between 0 and 99 to each token. For T_{CMP}, `LABEL` is one of two tokens randomly chosen from the vocab indicating whether $val(\texttt{TOK1}) < val(\texttt{TOK2})$. For T_{ADD}, `LABEL` is the token in $\mathcal{S}$ whose value equals $(val(\texttt{TOK1}) + val(\texttt{TOK2}))\ mod\ 100$. There are exactly 10K instances per task, which are split $50/50$ for training and testing. Critically, both tasks require models to learn certain geometrical structures to generalize [21]. This provides a measure for learning a task (as opposed to memorizing training instances) and a set of features to verify the interference hypothesis of Sec. 3. Below are a few instances from the comparison task `address analyze pony`, `resort zebrafish pony`, `cavities misconduct provisional`, where `pony` and `provisional` are the two label tokens that represent `True` and `False`.

A.4. Localizing Task Neurons

We conduct MLP ablation to identify layers that have the largest causal effects on the model output. The results are shown in Fig. 4. This aligns with our observation from the DAS localization experiment that first layer is the earliest layer where the global token order is causally encoded.

Appendix B. Proofs

B.1. Proof of Theorem 1

For fixed U, the task-k population loss is

$$\ell_k(U, D_k) = \mathbb{E}\big[\|\Lambda_k^{1/2} B_k^\top x - D_k U^\top x\|_2^2\big] \tag{2}$$

$$= \|\Lambda_k^{1/2} B_k^\top - D_k U^\top\|_F^2, \tag{3}$$

where the second identity uses $x \sim \mathcal{N}(0, I)$ and the standard relation $\mathbb{E}\|Ax\|_2^2 = \|A\|_F^2$. This is a linear least-squares problem in D_k, so the minimizer is

$$D_k^* = \Lambda_k^{1/2} B_k^\top U. \tag{4}$$

Substituting back gives

$$\ell_k(U) = \big\|\Lambda_k^{1/2} B_k^\top (I - P_U)\big\|_F^2 \tag{5}$$

$$= \mathrm{Tr}\Big(\Lambda_k^{1/2} B_k^\top (I - P_U) B_k \Lambda_k^{1/2}\Big) \tag{6}$$

$$= \mathrm{Tr}\big((I - P_U) C_k\big). \tag{7}$$

Summing with weights π_k yields

$$L_N(U) = \sum_{k=1}^K \pi_k \ell_k(U) = \mathrm{Tr}(M) - \mathrm{Tr}(U^\top M U), \qquad M := \sum_{k=1}^K \pi_k C_k. \tag{8}$$

Because M is symmetric positive semidefinite with finite trace, minimizing $L_N(U)$ is equivalent to maximizing $\mathrm{Tr}(U^\top M U)$ over all orthonormal U. By Ky Fan's variational principle,

$$\max_{U^\top U = I_N} \mathrm{Tr}(U^\top M U) = \sum_{i=1}^N \mu_i, \tag{9}$$

where $\mu_1 \geq \mu_2 \geq \cdots$ are the eigenvalues of M. Therefore any minimizer spans the top-N eigenspace of M and the optimal loss is

$$L_N^* = \mathrm{Tr}(M) - \sum_{i=1}^N \mu_i = \sum_{i>N} \mu_i. \tag{10}$$

In the orthogonal-block model,

$$M = \sum_{k,j} \pi_k \lambda_{k,j}\, b_{k,j} b_{k,j}^\top, \tag{11}$$

so the vectors $b_{k,j}$ are eigenvectors of M with eigenvalues $u_{k,j} = \pi_k \lambda_{k,j}$. Thus the width-$N$ optimum keeps the N largest utilities, up to arbitrary tie-breaking at the cutoff. If task k contributes $n_k(N)$ retained coordinates, then its residual loss is

$$\ell_k^*(N) = \sum_{j > n_k(N)} \lambda_{k,j}. \tag{12}$$

B.2. Expected-flow equilibria

Proposition 5 (Expected-flow equilibria) *If stochastic task sampling is replaced by its expectation under the mixture, the encoder follows the gradient flow*

$$\dot{U} = 2(I - P_U)MU. \tag{13}$$

A point U is stationary if and only if $(I - P_U)MU = 0$, equivalently if $\mathrm{span}(U)$ is an invariant subspace of M. If $\mu_N(M) > \mu_{N+1}(M)$, the top-N eigenspace is the unique asymptotically stable stationary subspace. If $N \geq \mathrm{rank}(M)$, then every U with $\mathrm{span}(M) \subseteq \mathrm{span}(U)$ is both stationary and globally optimal.

Proof From Theorem 1, the objective on the Stiefel manifold is $L_N(U) = \mathrm{Tr}(M) - \mathrm{Tr}(U^\top MU)$. Its Euclidean gradient is $-2MU$, and projecting onto the tangent space of the Stiefel manifold gives the Riemannian gradient $-2(I - P_U)MU$. Gradient descent therefore follows $\dot{U} = 2(I - P_U)MU$. Stationarity is exactly the condition $(I - P_U)MU = 0$, which means MU lies in the span of U; equivalently, $\mathrm{span}(U)$ is M-invariant. If $\mu_N(M) > \mu_{N+1}(M)$, the top-N invariant subspace is isolated and is the unique asymptotically stable principal subspace under this flow. If $N \geq \mathrm{rank}(M)$ and $\mathrm{span}(M) \subseteq \mathrm{span}(U)$, then $P_U M = M$, so $(I - P_U)MU = 0$ and $L_N(U) = 0$. ■

B.3. Proof of Theorem 2

Write

$$G_{\mathsf{F}}(U) = 2(I - P_U)M_{\mathsf{F}}^{1/2} M_{\mathsf{F}}^{1/2} U. \tag{14}$$

Using $\|AB\|_F \leq \|A\|_F \|B\|_{\mathrm{op}}$ gives

$$\|G_{\mathsf{F}}(U)\|_F \leq 2\,\|(I - P_U)M_{\mathsf{F}}^{1/2}\|_F\,\|M_{\mathsf{F}}^{1/2}U\|_{\mathrm{op}} \tag{15}$$

$$\leq 2\sqrt{\mathrm{Tr}\big((I - P_U)M_{\mathsf{F}}\big)}\,\sqrt{\lambda_1(M_{\mathsf{F}})} \tag{16}$$

$$= 2\sqrt{\lambda_1(M_{\mathsf{F}})\,\delta_{\mathsf{F}}(U)}. \tag{17}$$

This proves the bound. Note that only the forward implication is used in the main paper: small residual common-task signal implies weak aggregate common-task drift. The converse need not hold for arbitrary invariant but suboptimal subspaces.

B.4. Exact disappearance of common-task interference in finite rank

B.5. Proof of Proposition 4

Let U denote the common-task width-N solution and let u_i be one occupied common eigenvector with eigenvalue μ_i^{F}. Replace that vector by

$$v_i(\theta) = \cos\theta\, u_i + \sin\theta\, b_r, \tag{18}$$

while keeping the remaining $N - 1$ directions fixed. Because u_i is an eigenvector of M_{F} and b_r is orthogonal to the common block, the contribution of this one direction to the objective $\mathrm{Tr}(P_U M)$ is

$$\langle v_i(\theta), M v_i(\theta)\rangle = \mu_i^{\mathsf{F}} \cos^2\theta + \pi_r \lambda_r \sin^2\theta. \tag{19}$$

Subtracting the value at $\theta = 0$ gives

$$\Delta\,\mathrm{Tr}(P_U M) = (\pi_r \lambda_r - \mu_i^{\mathsf{F}})\sin^2\theta. \tag{20}$$

Since $L = \mathrm{Tr}(M) - \mathrm{Tr}(P_U M)$, the loss change is

$$\Delta L = (\mu_i^{\mathsf{F}} - \pi_r \lambda_r) \sin^2 \theta. \tag{21}$$

Hence perturbations toward b_r decrease the loss if and only if $\pi_r \lambda_r > \mu_i^{\mathsf{F}}$. The rare feature invades first through the weakest occupied common direction, which has curvature μ_N^{F}, proving the stated threshold.

B.6. Microscopic competition in a one-neuron, two-task model

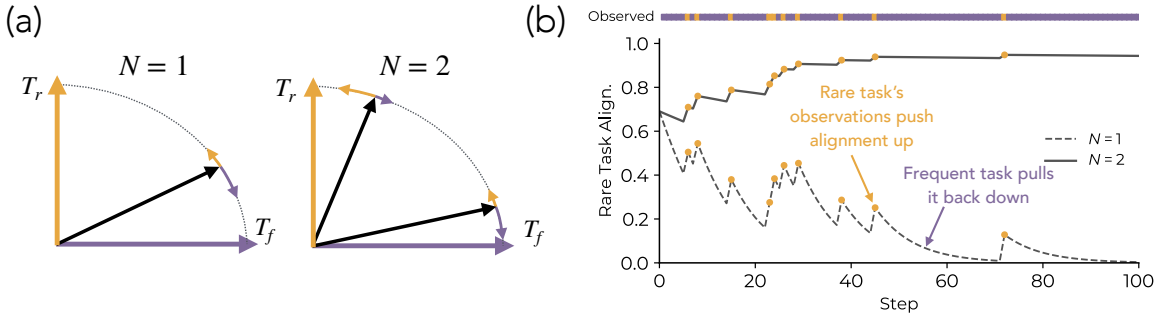

Figure 5: **Competition Dynamics over Neurons.** Rare task alignment over training for a softmax-gated model of 1 vs. 2 neurons. (a) Two orthogonal task directions T_f (frequent, sampled with probability 0.9) and T_r (rare, probability 0.1) compete for neurons. (b) With a single neuron, the frequent task dominates; with two neurons, one neuron specializes to each task, allowing rare task alignment to reach and sustain values near 1.

Example 1 (One neuron, two orthogonal tasks) *Let $a, b \in \mathbb{R}^d$ be orthonormal and consider rank-one tasks with covariances $C_a = aa^\top$ and $C_b = bb^\top$. A width-1 encoder is a unit vector*

$$u = \cos\theta\, a + \sin\theta\, b. \tag{22}$$

The task losses are

$$\ell_a(u) = \sin^2\theta, \qquad \ell_b(u) = \cos^2\theta. \tag{23}$$

A gradient step on task a obeys $\theta^+ = \theta - \eta \sin(2\theta) + O(\eta^2)$, while a step on task b obeys $\theta^+ = \theta + \eta \sin(2\theta) + O(\eta^2)$. If task a appears with probability p and task b with probability $q < p$, then

$$\mathbb{E}[\Delta\theta \mid \theta] = \eta(q - p)\sin(2\theta) + O(\eta^2), \tag{24}$$

which drives the neuron toward the common task. Near $\theta = 0$, if a rare-task update is followed by G common-task updates, then

$$\theta_G \approx (1 - 2\eta)^G \theta_0 \approx e^{-2\eta G}\theta_0. \tag{25}$$

Thus rare-task alignment decays exponentially across the gap between rare observations.

Proof The loss identities follow from $u^\top aa^\top u = \cos^2\theta$ and $u^\top bb^\top u = \sin^2\theta$. Differentiating yields

$$\frac{d}{d\theta}\sin^2\theta = \sin(2\theta), \qquad \frac{d}{d\theta}\cos^2\theta = -\sin(2\theta), \tag{26}$$

which gives the stated updates under gradient descent. Taking the expectation under the task mixture yields the drift formula. Linearizing $\sin(2\theta) \approx 2\theta$ near zero gives the exponential decay estimate. ■

The dynamics posited above are also exemplified in Fig. 5.

Appendix C. Further Experimental Results: Complexity Sweeps

In the main paper, we kept "complexity", i.e., the number of directions used for defining the target variable constant across tasks; specifically, tasks in the main paper require 5 directions to cover 90% energy in the task spectrum (i.e., solving for r in $\arg\min_r \frac{\sum_{j=1}^{j=r} \lambda_{k,j}}{\sum_j \lambda_{k,j}} > 0.90$ gives $r = 5$). In this section, we vary this property by changing the power-law coefficient underlying the task spectrum, i.e., since $\lambda_{k,j} \propto j^{-\alpha_k}$, we vary the range of α_k across tasks. We define ranges of $[\alpha_{\text{min}}, \alpha_{\text{max}}]$, split the range uniformly into K values, and assign the k^{th} value to α_k. The most frequent task is assigned the value α_{max}, giving it the fastest decaying spectrum and hence making it the simplest task, while the rarest task is assigned the value α_{min}, giving it the slower decaying spectrum and making it most complex. In particular, we choose ranges (see Fig. 6) such that the task complexity varies between $[4, 7]$ and $[2, 12]$ across $K = 32$ tasks.

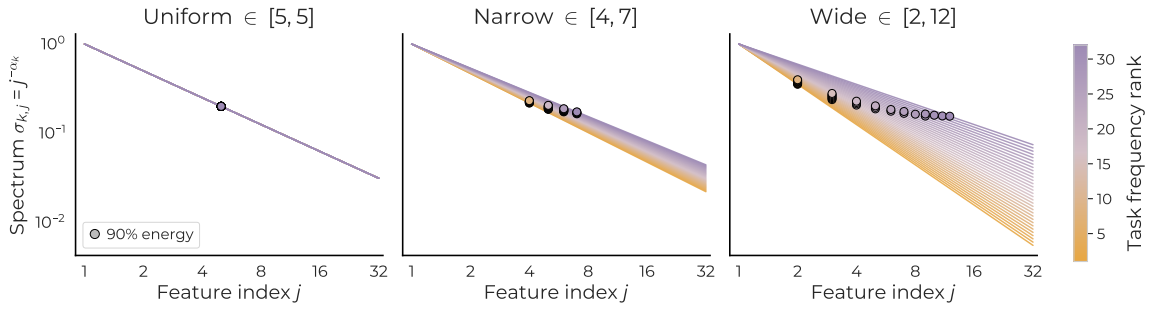

Figure 6: **Task Spectra.** We use power-law task spectra to vary the complexity of a task in our experiments, i.e., the j^{th} feature contributes signal proportional to $j^{-\alpha_k}$ for the k^{th} task. While the main paper studies the setting with uniform values for α_k, hence making frequency the core knob for varying utility, we now vary complexity by splitting a range of α values; this results in task spectra such that number of directions to cover 90% of task signal now takes 4–7 directions for the "narrow" range scenario, while $2 - -12$ directions for the wider range scenario.

C.1. Feature Utility Predicts Learning Order

We first reproduce Fig. 1. We split the figure into two parts, showing the critical width boundary as a function of task frequency in heatmaps in Fig. 7 and the per-task loss predictability based on feature utilities in Fig. 8; for reference, we include our baseline results from the main paper, where the task spectra were uniform.

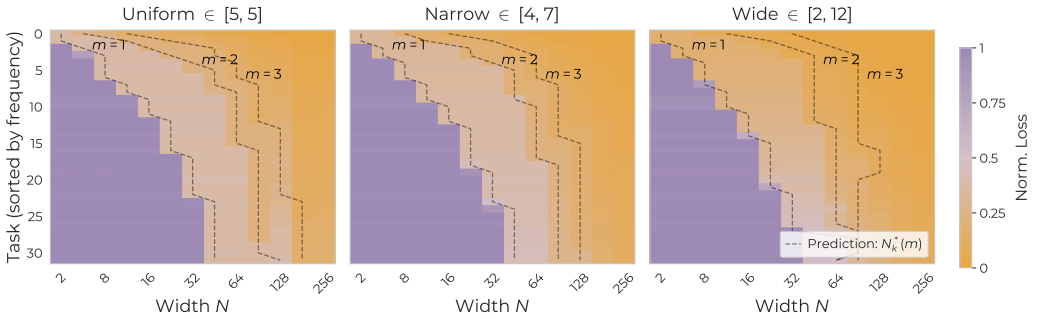

Figure 7: **Learning Phases Under Varying Complexity.** Reproduction of Fig. 1a under varying task spectra. We see increase in the complexity gap leads to higher emphasis on the top two modes' learning, since under a power-law spectrum decay, the eigenvalue associated with larger modes will be small. More critically, learning order is now not monotonically predicted by frequency alone: this is most easily visible in the results for wide complexity range scenario, where we see the "most complex" task's third mode is in fact high enough utility to get learned before more frequent task's higher order modes, resulting in a non-monotonic boundary.

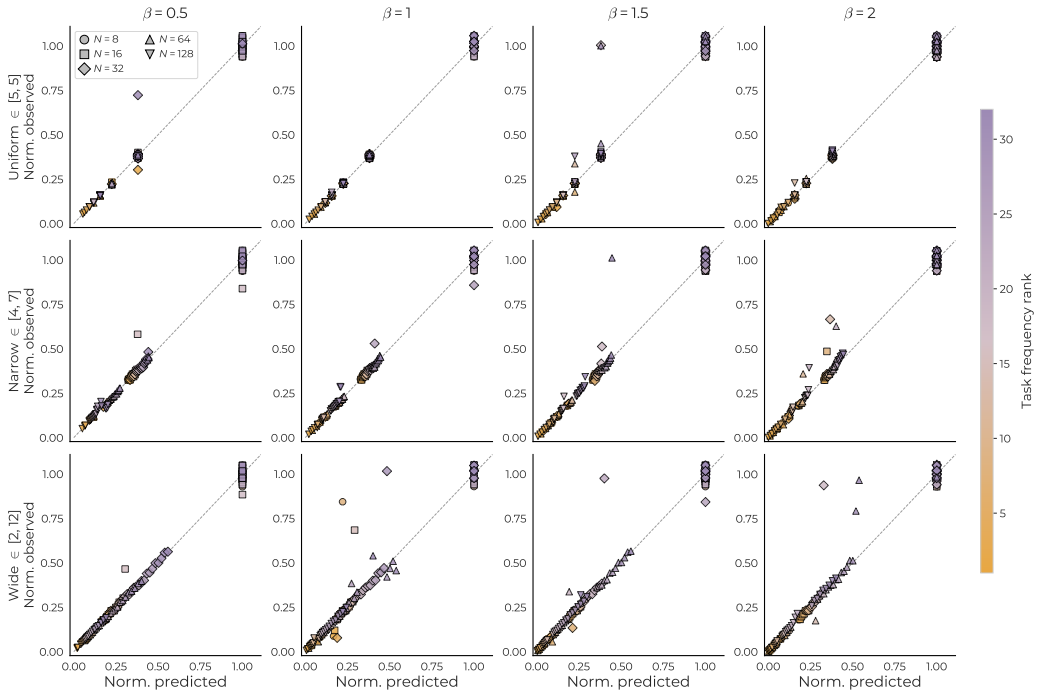

Figure 8: **Feature Utilities Continue to Predict Learning.** Reproduction of Fig. 1b under varying task spectra. While Fig. 7 shows frequency, by itself, is insufficient to predict learning of a task, the current plot shows the empirically observed loss and the loss derived out of the assumption that N neurons will learn top N utility features continues to align well. This confirms the learning dynamic in the non-uniform complexity scenarios requires accounting for both frequency and complexity: higher-frequency tasks *may be learned after* a lower-frequency task if the complexity of former is more than the latter.

C.2. Competition Dynamics Disallow Learning of Most Rare and Complex Task

Our results above showed that learning trends under varying task complexity are modulated by both task frequency and complexity, as predicted by our account in Sec. 3. We now show the competition dynamics picture posited in that section continues to follow in these settings as well. In particular, we plot the learning of the top-3 most frequent tasks (measured by normalized signal; see Sec. 3 for details) and the rare-most task as a function of residual, i.e., signal remaining to be explained in the frequent tasks. As shown in Fig. 9, the critical width predicted to be necessary for learning of the rare-most task, by rendering the residual sufficiently small for most frequent tasks, continues to hold true in this setup as well.

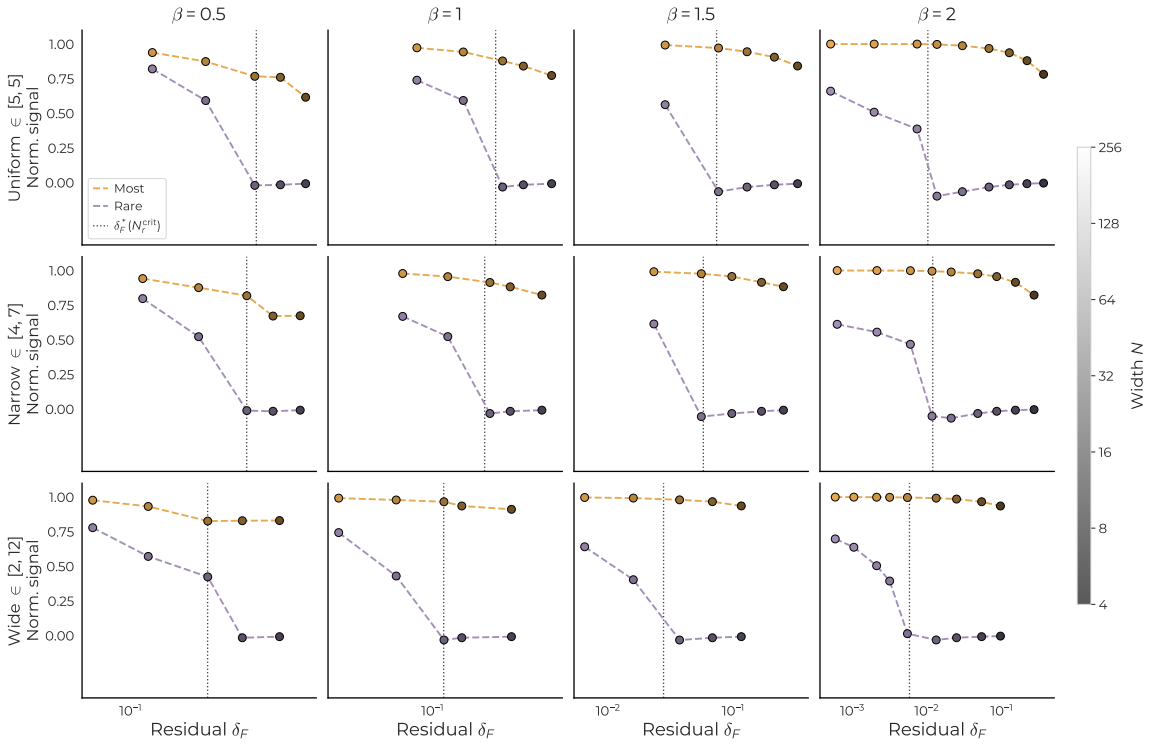

Figure 9: **Complexity Residual.** We plot the amount of signal encoded in model representation for most frequent and rarest tasks as a function of width N and remaining residual δ_{F}. Inline with our predictions, we see larger models perfectly capture tasks of all frequencies, while smaller models do not. Meanwhile, even for the largest models, when the residual signal remaining to explain for frequent tasks is high, rarer tasks struggle to be learned.

C.3. Reduced Interference Aids Learning of Most Rare and Complex Task

We now validate our argument for how data-centric bottlenecks, i.e., the low-frequency and high-complexity nature of a task, is circumvented by a larger model: by virtue of having more parameters, a larger model witnesses reduced interference over per-task gradients. To this end, we redo the batch-injection experiments from Fig. 2 and plot the retention dynamics for the lowest aggregate utility task across settings. Results are shown in Figs. 10. We see similar results as before: larger models show better retention of observed signal from a task, allowing them to bootstrap on these past

observations and eventually learn the task. Meanwhile, a medium width model is able to do so only when the task is observed sufficiently frequently, i.e., the gap is low. Comparing with the case when the width is too low, we see the model never learns the task and the retention dynamics concretely show why: the model is unable to retain signal for the observed task for long enough.

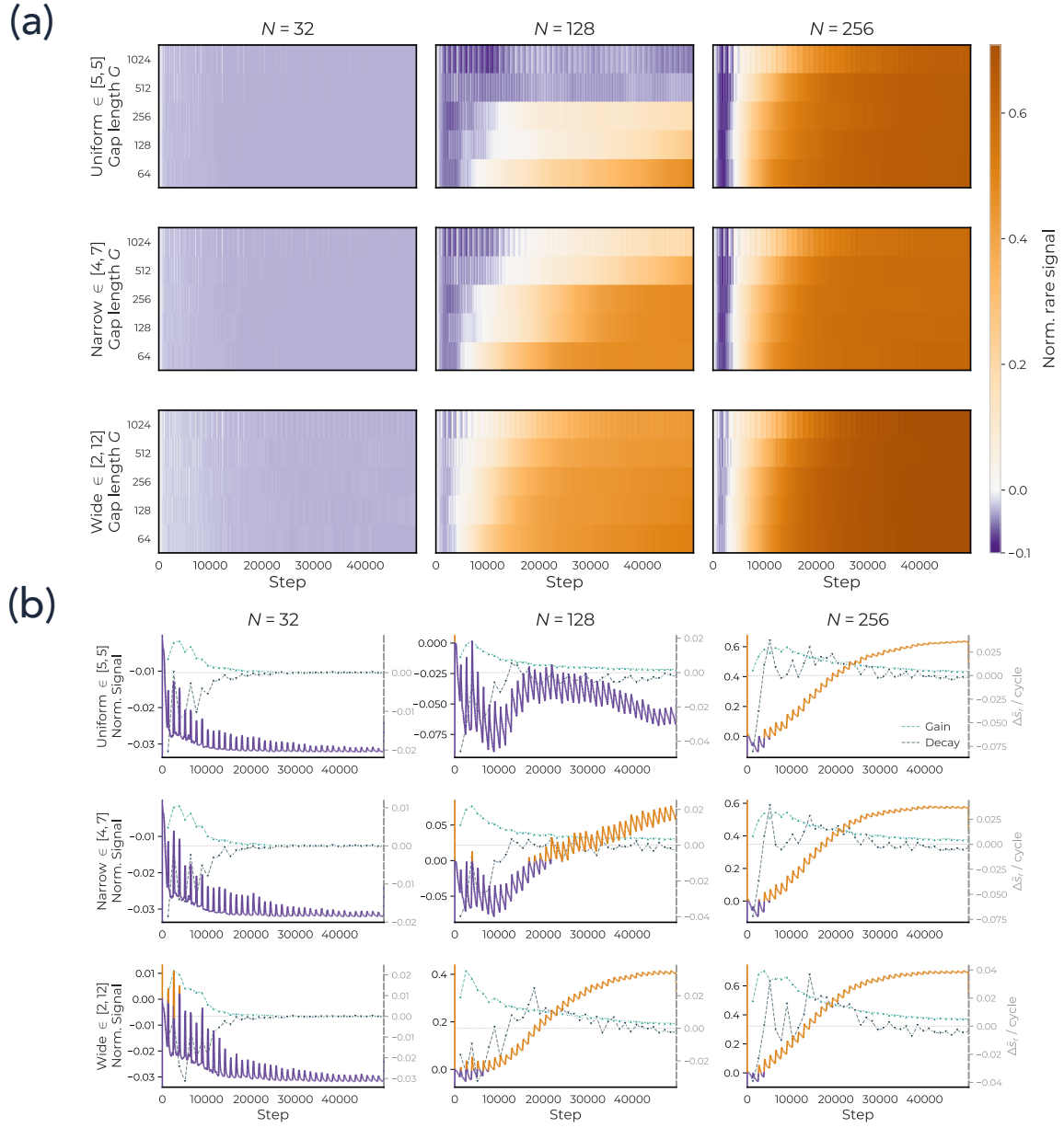

Figure 10: **Complexity Retention Phases.** We isolate retention by training with a matched-frequency injection protocol: the lowest-total utility task is withheld for G steps and then reintroduced in a batch such that its overall frequency is consistent across settings. (a) Training dynamics for $G = 1280$. We see small models briefly encode the rare task (Norm. signal $\tilde{s_r}$: left-y axis) after each injection; specifically, $\Delta\tilde{s_r}$ increases at point of injection, as shown by green dotted line ('gain'). However, as frequent-task updates resume, this signal is quickly lost ('decay': gray dotted line). Meanwhile, larger models retain more of the rare-task signal between injections and accumulate it over training. (b) Across injection gaps G and widths N, rare-task signal decays rapidly in narrow models but remains stable in wider models, while frequent-task signal is largely unaffected. These results support the reduced-interference mechanism: scaling provides enough representational capacity that updates from frequent tasks no longer overwrite rare-task features before the next rare observation arrives.

Appendix D. Further Experimental Results: Frequency Sweeps

D.1. Features and Tasks are Learned in Order of Utility

D.1.1. EXTENDING PHASE DIAGRAM

We sweep the power-law exponent β defining the task prior by varying it over $\{0.5, 1.0, 1.5, 2.0\}$. Unlike Fig. 1, we only sweep 5 values of widths—specifically, $N \in \{8, 16, 32, 64, 128\}$. Correspondingly, the staircase is rendered at lower resolution.

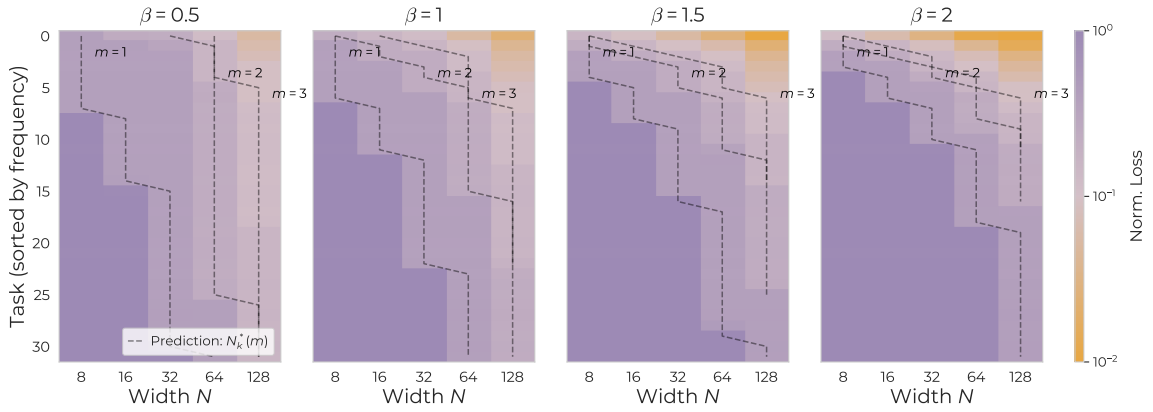

Figure 11: **Feature Utility Predicts Order of Learning.** We extend results from Fig. 1 by analyzing values $\beta \in \{0.5, 1.0, 1.5, 2.0\}$. Each panel reports the normalized per-task loss, i.e., $\ell_k(N)/\ell_{k,\text{baseline}}$ as a function of width N. Tasks are sorted top-to-bottom by descending prior frequency so the most-frequent task occupies the top row. Dashed staircases are the theoretical thresholds $N_k^*(m)$ for $m = 1, 2, 3$ computed from the per-direction utility ordering of Theorem 1. The empirical learned region (orange) tracks the $m = 1$ staircase across all four prior skews; deeper-orange cells in the steeper-prior panels ($\beta = 1.5, 2$) reflect the model spending its width budget on additional directions of the leading tasks rather than on rarer tasks, in agreement with the account posited in the main paper for how scaling interacts with data properties.

D.1.2. SIMPLIFIED CASE: RANK-1 TASKS

Theorem 1 predicts that a width-N minimizer retains the N task-features with largest utility $u_{kj} = \pi_k \lambda_{kj}$. We obtain a sharp quantitative test of this claim by collapsing the orthogonal-block setup to its rank-1 specialization: setting $d_T = 1$ and $\alpha_k = 1$ makes every task rank-1 with $\lambda_k = 1$, so the utility ordering reduces to the prior ordering, and the predicted critical width for task k becomes

$$N_{\text{crit}}(k) \;=\; \#\{j \neq k : \pi_j \lambda_j > \pi_k \lambda_k\} \;=\; k, \tag{27}$$

i.e., a perfectly linear staircase in task index.

Setup. We train the linear-bottleneck student described in Sec. 3 on a mixture of $K = 32$ rank-1 orthogonal tasks, ambient dimension $D = 1024$, and a power-law prior with exponent $\beta = 2$. We sweep the encoder width $N \in \{1, 2, 3, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 64\}$ and read out the per-task subspace alignment $s_k(U) = \|P_U b_k\|^2$ at the end of training (rank-1 specialization of the per-task signal of Sec. 3, so $s_k(U)$ lies between N/D at random initialization and 1 when b_k is fully captured by the encoder subspace).

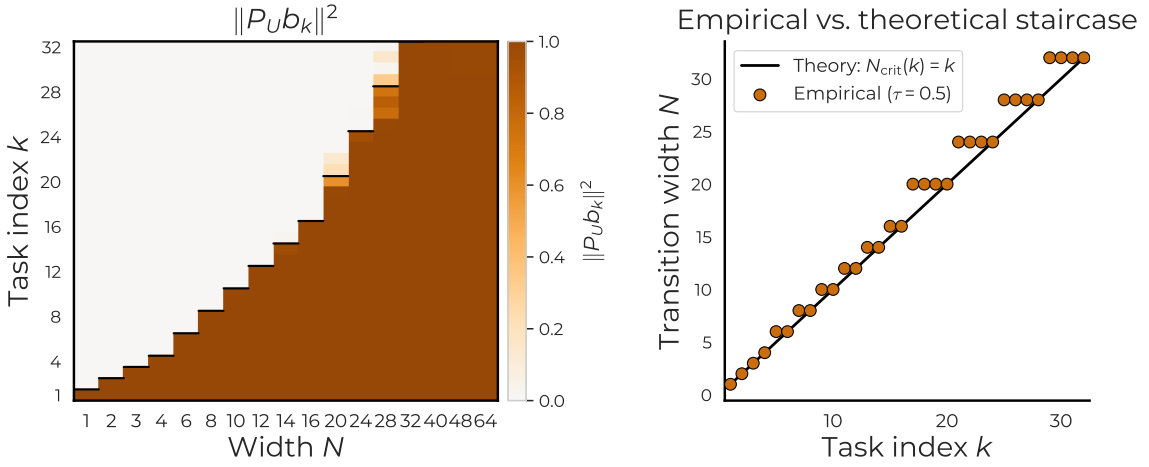

Figure 12: **Rank-1 Verification of Utility Predicting Learning Order. Left:** Per-task subspace alignment $\|P_U b_k\|^2$ at the end of training as a function of width N and task index k. By our account, we expect tasks $1..N$ be retained, while tasks $N+1..K$ to not be retained. The black step segments in the heatmap mark the predicted retention horizon $k = N$ per width. Results align well with our expectations. **Right:** Empirical transition width $N_{\text{emp}}(k)$ (markers) sits on the theoretical staircase $N_{\text{crit}}(k) = k$ (line) within the resolution of the width sweep. Plateaus at $k = 5, 7, 9, \ldots$ reflect the gap in sampled widths between adjacent grid points and are not deviations from theory.

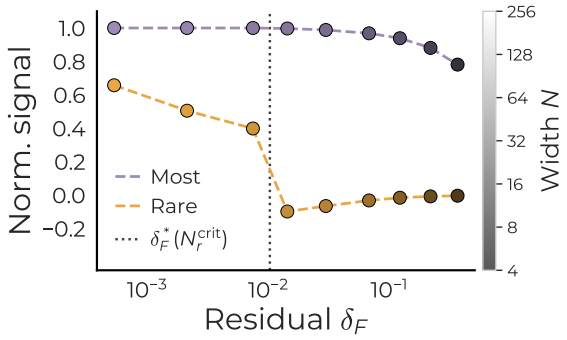

Figure 13: **Residual Controls Learning.** We plot signals encoded in model representations for most frequent and rarest tasks as a function of width N and remaining residual δ_{F}. Inline with our predictions, we see larger models perfectly capture tasks of all frequencies, while smaller models do not. Meanwhile, even for the largest models, when the residual signal remaining to explain for frequent tasks is high, rarer tasks struggle to be learned.

Result. See Figure 12. We overlay the empirical transition width $N_{\text{emp}}(k) = \min\{N \in \text{grid} : s_k(U) > 0.5\}$ on the theoretical staircase (27), finding almost perfect alignment (minimal disparities are an artifact of the sampled width grid).

D.2. Residual Controls Learning

We next provide further validation for results in Fig. 13 by repeating experiments across different values of exponents in the power-law task prior. Specifically we vary β from the set

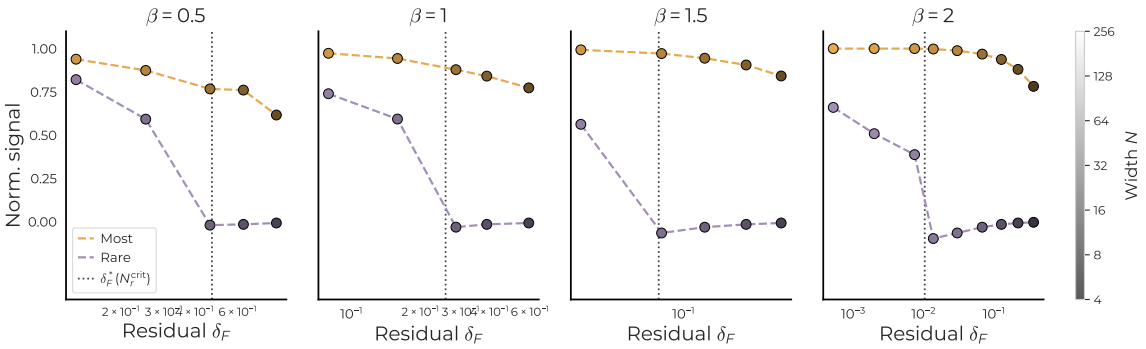

Figure 14: **Residual Controls Rare-Task Learning**. We vary $\beta \in \{0.5, 1.0, 1.5, 2.0\}$. Each panel reports the normalized rare-task and most-frequent-task signal as a function of the frequent-task residual δ_F, with width N encoded by marker brightness (dark = small N, bright = large N; see grayscale colorbar on the right). Dashed vertical line marks the analytic threshold $\delta_F^*(N_r^{\text{crit}})$ computed from Corollary 3 under that panel's β. The two-phase dynamic predicted by Theorem 2 and seen in Fig. 13 is preserved across all four prior skews; the shift in the threshold's location with β is exactly what theory predicts.

$\{0.5, 1.0, 1.5, 2.0\}$. For each run we compute the per-task signal $s_k(U) = \text{Tr}(P_U C_k)/\text{Tr}(C_k)$, i.e., how well the model representation encodes information about the k^{th} task, and the residual $\delta_F(U) = \sum_{k \in F} \pi_k \left(1 - s_k(U)\right) \|a_k\|^2$ from the final checkpoint. The frequent set F is defined as the smallest set of tasks whose cumulative mass meets 0.8; under $\beta \in \{0.5, 1, 1.5, 2\}$ this yields $|F| = \{6, 3, 2, 2\}$ respectively, reflecting how a flatter prior spreads the loss budget across more frequent tasks. Results are reported in Figure 14. We see a precise kink similar to Fig. 13 when rare-task signal drops to zero once δ_F exceeds the analytic threshold $\delta_F^*(N_r^{\text{crit}})$, and rises steeply to near-unity once δ_F falls below it. We also see the threshold itself shifts left as β grows: a steeper prior makes the rare task's leading utility $\pi_r \lambda_r$ much smaller, so a smaller residual is required to "free up" encoder directions that can then capture the rare task.

D.3. Per-gap dynamics: Reproducing retention results across different injection gaps and widths

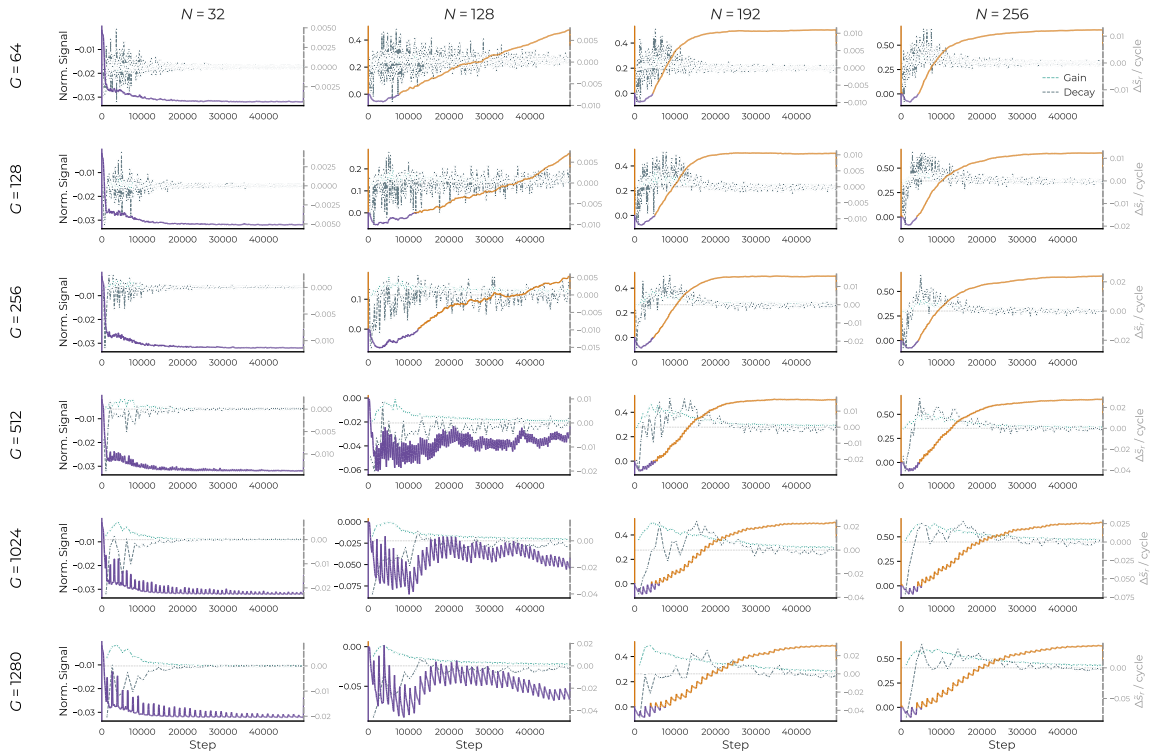

Figure 15: **Per-gap dynamics: Reproducing retention results across different injection gaps and widths.** We vary the injection gap G in the set $\{64, 128, 256, 512, 1024, 1280\}$ (top to bottom) and reproduce the results shown in Fig. 2a for widths $N \in \{32, 96, 128, 192, 256\}$. In each cell, the left y-axis reports the normalized rare-task signal $\tilde{s}_r(U_t)$, while the right axis (gray) is the gain / decay curves reporting how much the signal for rare task grows vs. decays as a function of time. We see analogous results as the main paper: larger models retain and preserve the learned signal, while smaller models require the gaps to be sufficiently small if learning is to occur at all.

We reproduce results from Fig. 2a by reporting the joint dynamics of the normalized rare-task signal $\tilde{s}_r(U_t)$ and its gain / decay dynamics as a function of rare-task injection events. Similar to results seen in the main paper, we find larger models retain and preserve the learned signal, while smaller models require the gaps to be sufficiently small if learning is to occur at all.

D.4. Effects of Scaling Data: Learning Bottleneck Persists at Long Training Horizon

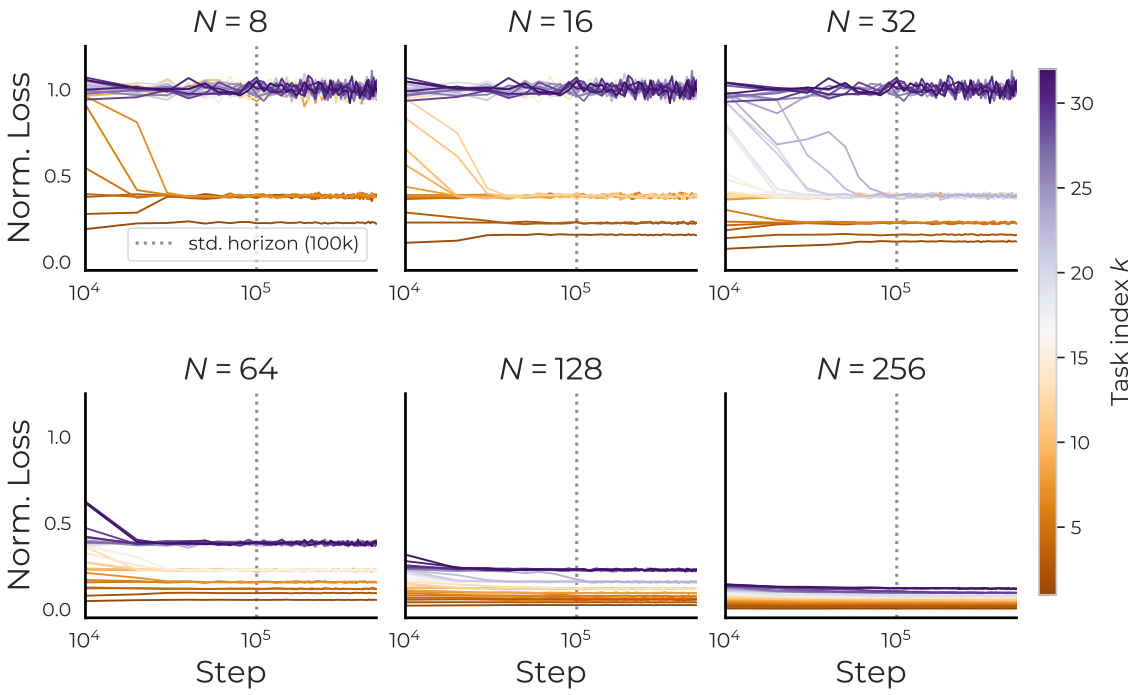

Figure 16: **Persistence of the multi-rank phase diagram at 1M steps.** Per-task normalized loss $\ell_k/\ell_{k,\text{baseline}}$ versus training step (log-x, linear-y) for six widths $N \in \{8, 16, 32, 64, 128, 256\}$; $\ell_{k,\text{baseline}} = \|a_k\|^2/D_t$ is the mean-predictor MSE per task. Tasks colored by index from orange ($k = 1$, most frequent) to purple ($k = 32$, rarest); vertical dotted line marks the training budget used in main paper, i.e., 100K steps. We clearly see that at every width, tasks that can fit model capacity (top-by-utility) drop near zero by the standard horizon and stay there; above-capacity tasks remain near the mean-predictor baseline and do not bend downward across longer training.

In the main paper, especially Sec. 2, we distinguish between finite vs. asymptotic training. However, most of our training runs use a budget of 100K training iterations. To contextualize that this budget is sufficient for the claims made in the paper, we extend training runs to 1M steps for 6 values of model widths ($N = \{8, 16, 32, 64, 128, 256\}$, subsampling the range of widths analyzed in the main paper; the setup remains the same otherwise as Fig. 1. We find that results (see Fig. 16) are stable at *much* longer horizons: above-capacity tasks do not slowly close the gap given more training; instead they remain at or below the random-projection baseline indefinitely.

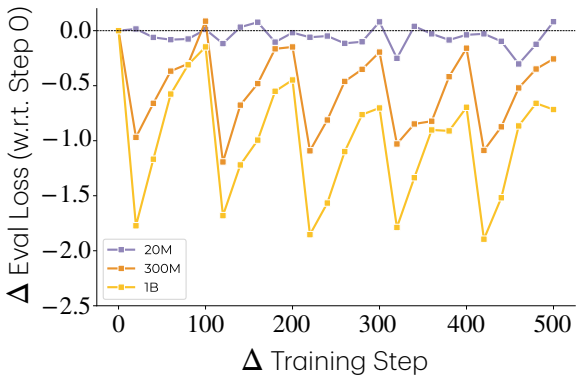

Figure 17: **Rare-Task Retention.** Larger models can retain the injected task information better, i.e., larger task eval loss drop, when injecting task instances every 100 batches.

Appendix E. OLMo Gradient Evidence

We now analyze how task gradients interfere with non-task gradients on a set of neurons that implement the task circuit. We focus on T_{CMP} training runs in Fig. 17, where 100 task instances are injected every 100 steps.

Task neurons. We first identify which MLP layers implement the task features. For all the models that we compared, the first layer MLP has the largest causal effects on task predictions. We further identify the top K neurons in the first layer MLP that have the largest gradient magnitude and use the gradients of these neurons for analysis. Details can be found in Appendix A.4.

Task reference direction g_r. We estimate the task reference direction using the aggregated gradient of the task loss computed over all 10K task instances, an analogy to G_r in the toy setting. This direction may shift across training steps, however, at each step, it is the optimal task direction.

Larger models have less gradient interference between general language modeling task and the injected task. We quantify the relation between the task reference g_r and the batch gradient g, which can be further decomposed into gradient from the task tokens g_t (if exists in batch) and non-task tokens g_{nt}, i.e., $g = g_t + g_{nt}$. We fist measure the cosine similarity between task reference and batch gradient direction, replicating the results in Fig. 2. We additionally analyze whether task or non-task tokens contribute to this similarity; while task token gradient aligning with task reference g_r is expected, non-task token gradient with non-zero cosine similarity suggests that the language modeling direction is interfering with the task gradient direction.

Results are shown in Fig. 3. In the top panel, larger models have higher similarity between g and g_r at the injection steps, 0.08 ± 0.02 for the 1B model and 0.04 ± 0.04 for the 300M model, the similarity typically regresses towards zero between injections. For the 20M model, the similarity scores oscillate wildly across batches, even at the injection step. In fact, the high similarity between non-task gradient g_{nt} and g_r reveals that for the 20M model the batch gradient similarity mostly comes from randomly collisions with task direction, with a similarity score of 0.10 ± 0.09, while for larger models, g_{nt} is almost orthogonal to g_r, with $7.58 \times 10^{-5} \pm 0.02$ for the 1B model, suggesting little to no gradient interference on this set of neurons.