

Quiet Feature Learning in Algorithmic Tasks

Prudhviraj Naidu, Zixian Wang, Leon Bergen*, Ramamohan Paturi*

Department of Computer Science
UC San Diego

San Diego, CA 92093, USA

prnaidu@ucsd.edu, ziw081@ucsd.edu, lbergen@ucsd.edu, rpaturi@ucsd.edu

Abstract

We train Transformer-based language models on ten foundational algorithmic tasks and observe pronounced *phase transitions* in their loss curves that deviate from established power-law scaling trends. Over large ranges of compute, the validation loss barely improves, then abruptly decreases. Probing the models’ internal representations reveals that *quiet features* are learned prior to any decrease in task loss. These quiet features represent intermediate algorithmic computations that do not by themselves improve the output loss. Ablation experiments demonstrate that individual quiet features are causally necessary for task performance. Our results demonstrate that substantial representational progress can remain hidden beneath an apparently flat loss curve, challenging the prevailing use of cross-entropy as a proxy for learning and motivating richer diagnostics for monitoring model training.

Code — <https://github.com/prudhvirajn/quiet-feature-learning-in-algorithmic-tasks>

Extended version — <https://arxiv.org/abs/2505.03997>

1 Introduction

Understanding how and when large language models acquire new capabilities has become an important question in deep learning. While language models demonstrated remarkable performance across a broad range of tasks, the precise mechanisms driving their improvements remain unknown. Recent discussions of “emergent abilities” – where larger-scale models outperform baselines abruptly, even though smaller-scale counterparts exhibit little improvement – have led to debate over whether such phenomena are genuine or artifacts of measurement (Wei et al. 2022; Ganguli et al. 2022; Schaeffer, Miranda, and Koyejo 2023).

Questions about emergent abilities are closely tied to the observation of scaling laws in model training (Kaplan et al. 2020; Ruan, Maddison, and Hashimoto 2024; Henighan et al. 2020; Dubey et al. 2024; OpenAI 2023). These scaling laws typically show a smooth, power-law relationship between compute and model performance. However, most

empirical demonstrations of these laws derive from heterogeneous data and tasks, leaving open the possibility that “averaging out” many distinct learning behaviors masks more abrupt transitions that occur for individual skills or subtasks.

In order to better understand skill learning in a tractable setting, we focus on ten foundational algorithmic problems spanning various input types. These algorithmic tasks have precisely defined solutions, making it straightforward to identify clear success criteria, isolate the specific features the model must learn, and ensure that improvements cannot be attributed to memorization or partial heuristics. These tasks allow us to investigate fine-grained learning phenomena which might otherwise be obscured by heterogeneous data.

Our key findings include:

1. **Phase transitions occur during learning:** We observe two distinct phases in scaling laws across tasks and input sizes. In the *slow phase*, loss improves minimally or remains flat. Then, loss drops rapidly (*fast phase*). We refer to the change between these two phases as a phase transition. Phase transitions occur for scaling laws estimated across many training runs and within individual training runs.
2. **Quiet features precede phase transitions:** Models learn meaningful internal representations during the slow phase, but these features do not yet yield noticeable gains in the output loss (we call these *quiet features*). Ablating them severely degrades performance, demonstrating they are causally related to the eventual sharp drop in loss.

These findings challenge the assumption that improvements in loss necessarily coincide with improvements in feature representations. Instead, substantive internal reorganization may occur below the surface, revealing itself only at discrete points during training.

The rest of this paper is organized as follows: Section 2 reviews related work in scaling laws, emergent abilities, and algorithmic learning. Section 3 describes our experimental methodology and presents our observations of phase transitions across tasks and input sizes. Section 4 introduces our feature analysis framework and demonstrates how quiet and loud features evolve during training. Finally, Section 5 discusses the broader implications of our findings and suggests directions for future research.

*These authors contributed equally.

Scaling Law Estimation

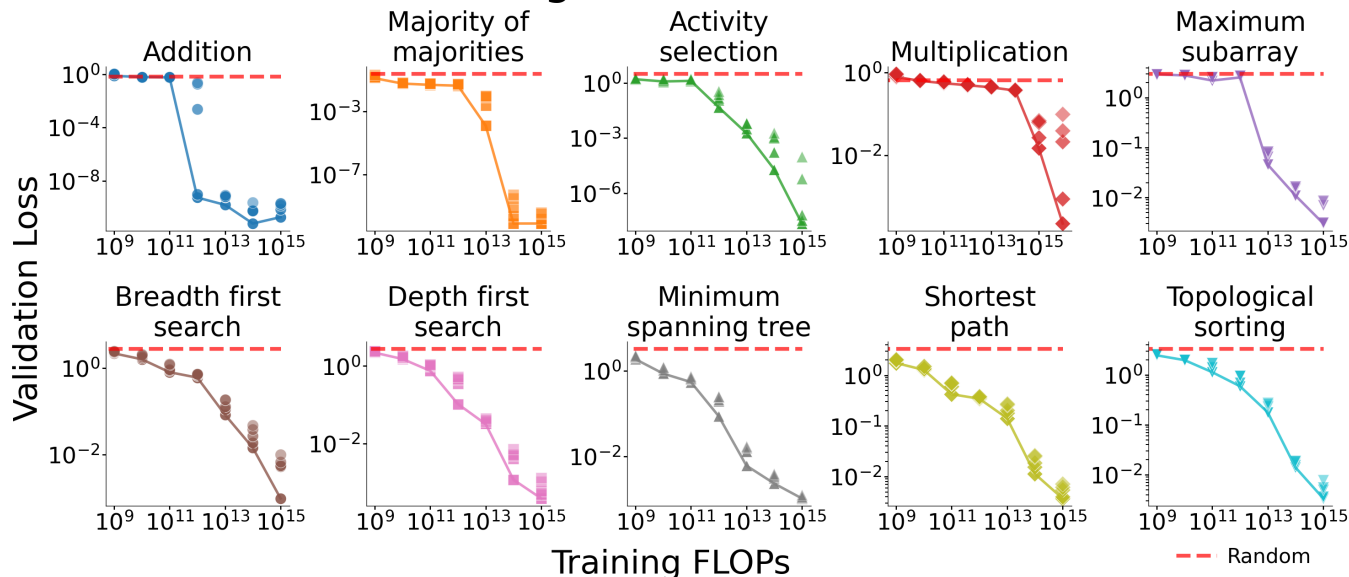


Figure 1: Model performance (validation loss) abruptly improves as we increase the model size, dataset size, and amount of compute (Training FLOPs) used for training. The input size for addition, multiplication and activity selection is 16. For graph tasks, the input size is 11. For maximum subarray, the input size is 64 while for majority of majorities it is 32. The red dotted line indicates random performance.

2 Related Work

Scaling Laws

Hestness et al. (2017) observed scaling dataset size and deep neural network model size led to a predictable decrease in generalization error for neural machine translation, language modeling, image classification and speech recognition. Kaplan et al. (2020) and Hoffmann et al. (2022) observed predictable relationships between training compute and language modeling loss. Henighan et al. (2020) extended this work for generative models across modalities: image, video, multimedia image-text and math. They demonstrated classification loss and error rates predictably decreased on downstream image classification tasks. Chen et al. (2021) studied language model performance on coding. They observed a predictable relationship between language modeling loss on a held out code corpus and model size.

Predicting LLM abilities

Ganguli et al. (2022) and Wei et al. (2022) demonstrated some large language models’ capabilities could not be predicted from capabilities of small language models. However, Schaeffer, Miranda, and Koyejo (2023), OpenAI (2023), Ruan, Maddison, and Hashimoto (2024), and Dubey et al. (2024) provide evidence that this is due to choice of metrics and that large language models capabilities can be predicted from small language models.

Proposed explanations for scaling laws

Michaud et al. (2023) proposed neural networks learn discrete skills called “quanta.” They argue that there is a strict

ordering, which they called Q sequence, in which quanta must be learned, and that the frequencies of these quanta follow a power law, leading to the power law relationship observed by Kaplan et al. (2020) and others. Hutter (2021) propose that the relationship between the error rate and dataset size is guided by the distribution of features in the data. They show a Zipfian distribution of features results in power law scaling.

Grokking

In grokking (Power et al. 2022; Nanda et al. 2023; Varma et al. 2023), a model trained for many epochs quickly memorizes the training set (thus achieving high training accuracy early) but only later learns a generalizing solution, causing a sudden jump in test accuracy. Our scaling law results are related to grokking, but occur in the single epoch setting. Unlike grokking, models trained in the single-epoch setting do not exhibit a transition from memorization to generalization.

Progress Measures

Several previous works have identified measures which track progress toward the final, fully-generalizing solution, even when the test loss shows no improvement. Barak et al. (2022) propose a metric for measuring similarity of network weights in the context of sparse parity, and demonstrate that this metric continuously improves throughout training, including prior to measurable improvement in generalization performance. Nanda et al. (2023) propose a different metric on network weights in the context of modular arithmetic

Single Training Runs

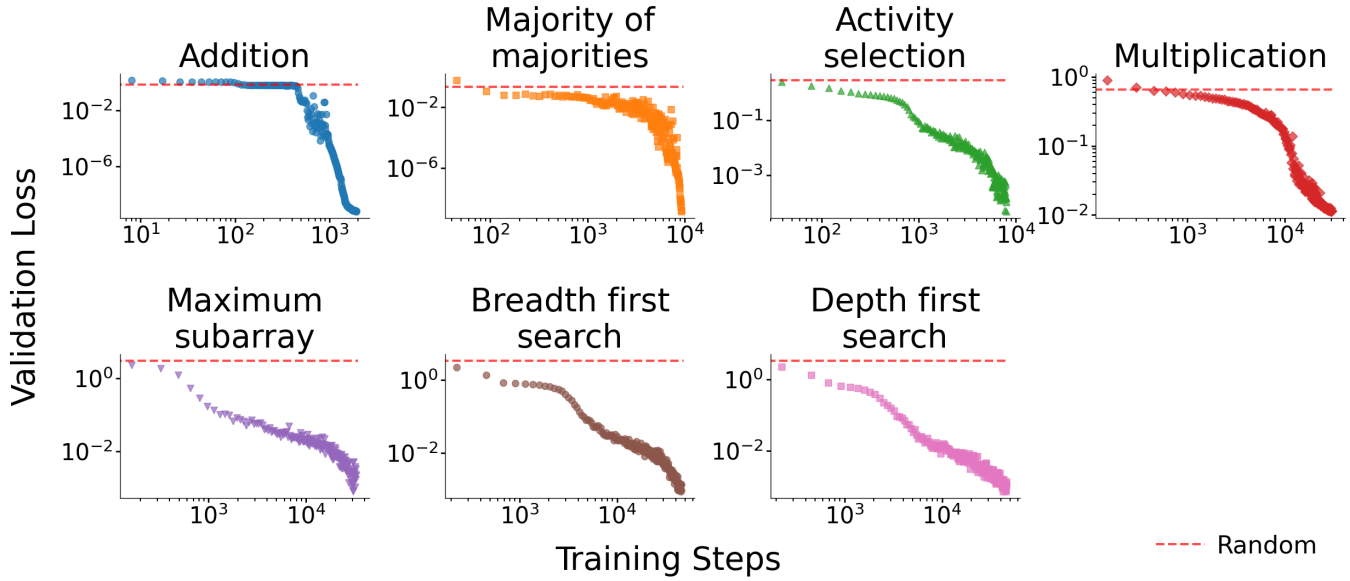


Figure 2: Model exhibit similar abrupt improvement in performance during a single training run. Plots show compute-optimal training runs for the smallest compute budget where test accuracy is 100%. The red dotted line indicates random performance.

grokking, and demonstrate that this metric improves before the phase transition. Mallinar et al. (2025) propose tracking features using Average Gradient Outer Product (AGOP) for Recursive Feature Machines. While this prior work has focused on measuring progress in model weights, they do not demonstrate that the networks are computing interpretable activations prior to generalization. We close this gap by directly probing for human-interpretable features and showing they appear well before the loss drop.

Phase Transitions

Phase transitions were previously observed for a limited number of algorithmic tasks. Olsson et al. (2022), Garg et al. (2022), and Edelman et al. (2024) find phase transitions for in-context learning during individual training runs. Barak et al. (2022) observed phase transitions in parity. Lee et al. (2024) measures relationships between test accuracy and number of examples (over a fixed model size), with observed phase transitions potentially being explained by the metric artifacts of Schaeffer, Miranda, and Koyejo (2023).

3 Scaling Laws for Algorithmic Tasks

We first aim to estimate scaling laws for 10 foundational algorithmic tasks. Scaling laws are estimated by training models over a range of compute budgets, and identifying the optimal model at each budget.

Task Formulation

We examine 10 algorithmic tasks which are drawn from three broad categories: binary arithmetic, graph algorithms and sequence-based optimization. The tasks capture a range

of input types, and have well-understood algorithms for solving them.

All tasks are formulated as sequence prediction problems. The input to the problem is serialized, and an autoregressive model is trained to predict the solution. All tasks use a standard cross-entropy loss, with the loss masked on the input tokens. We describe how we formulate three of the tasks below. For other tasks, please see Technical Appendix A.

Binary Addition We formulate n -bit binary addition as the following sequence prediction task:

$$x_1x_2 \dots x_n + y_1y_2 \dots y_n = z_1z_2 \dots z_{n+1} \langle \text{EOS} \rangle$$

where x , y , and z are binary numbers, presented from the least significant bit to the most significant bit. Each bit is represented as a separate token, and $+$, $=$, and $\langle \text{EOS} \rangle$ are also represented as individual tokens.

Breadth First Search Given a connected undirected graph G with n vertices $V = \{v_1, v_2, \dots, v_n\}$, a set of edges E , and a start vertex v_s , the task is to predict the traversal order in a breadth first search.

We formulate this as:

$$v_s v_{i_1} v_{j_1} \dots v_{i_m} v_{j_m} = v_{t_1} v_{t_2} \dots v_{t_n} \langle \text{EOS} \rangle$$

where (v_{i_k}, v_{j_k}) represents an edge in E , $m = |E|$ is the number of edges, and $v_{t_1} v_{t_2} \dots v_{t_n}$ is the complete BFS traversal sequence starting from v_s (where $v_{t_1} = v_s$). Ties in BFS ordering are broken by lexicographic ordering.

Maximum Subarray Given a sequence of n integers k_1, k_2, \dots, k_n where $k_i \in [-9, 9]$, the maximum subarray task is to predict the contiguous subarray with the maximum sum.

We formulate this as:

$$k_1 k_2 \dots k_n = k_i k_{i+1} \dots k_j \langle \text{EOS} \rangle$$

Where $k_i k_{i+1} \dots k_j$ is the maximum sum subarray ($i \leq j$).

Experimental Methodology

Model Training Each task is trained independently with the Transformer++ architecture. Transformer++ is a decoder-only transformer model with enhancements detailed in Technical Appendix Table 4, based on modifications in Llama and PALM (Gu and Dao 2024). This architecture is chosen because it has improved performance in scaling law experiments compared to other transformer variants (Gu and Dao 2024).

Models are trained with the AdamW optimizer (Loshchilov and Hutter 2019) with linear warmup followed with cosine learning rate annealing as prescribed by Hoffmann et al. (2022).

Estimating Scaling Laws The scaling law experiments aim to estimate the best performance achievable on a task given a compute budget. Separate scaling laws are estimated for each task and input size. Each model is trained up to a pre-specified compute budget, which ranged from $10^9 - 10^{15}$ FLOPs.¹ For each budget, we conduct a grid search across model sizes, batch sizes, and learning rates (see Technical Appendix Table 3 for details about the hyperparameter search). Following the procedure from Chinchilla (Hoffmann et al. 2022), the period of the learning rate scheduler is set to the number of training steps.

The number of training runs per task varies from 1316 to 3565, and the total number of training runs is 18544. All models are trained for at most a single epoch; each algorithmic task has a sufficient number of unique examples to avoid repetition even with the highest compute budgets. The number of training examples is determined based on training compute budget and model size, with all configurations evaluated using randomly generated validation and test sets with 1000 examples each. We choose the configuration with minimum validation loss for each training compute and designate it as the “compute-optimal validation loss.”

Scaling Law Results

We observe phase transitions for compute-optimal validation loss across three scenarios: (1) when we vary both model size & dataset size, (2) when we fix the model size & vary the dataset size, and (3) during individual (compute-optimal) training runs. Figure 1 shows that for six of the tasks, the compute-optimal validation loss undergoes a clear phase transition as the training compute budget increases. For these tasks, there are two distinct phases of learning: a slow phase and a fast phase. During the slow phase, loss is stagnant or decreasing slowly. During the fast phase, the loss decreases rapidly.

For addition, majority of majorities, activity selection and maximum subarray the validation loss is roughly constant

¹For multiplication, the maximum budget was increased to 10^{16} , since this was the minimum budget needed to train the task to 100% accuracy.

in the slow phase then suddenly goes to near zero during the fast phase. For multiplication and breadth first search, the slow phase has a gradual decrease followed by a steeper decrease in the fast phase.

Next, we investigate the effects of varying the dataset size. In Technical Appendix Figure 6, we fix the model size (selecting the model size corresponding to the smallest training compute budget that achieves 100% test accuracy) and increase the dataset size. We continue to observe phase transitions even when the model size is fixed. In this setting, additional graph tasks exhibit distinct phase transitions.

We next analyze model behavior within individual training runs.² Figure 2 shows these individual training runs exhibit phase transitions in the loss. For addition and majority of majorities, there is a predictable power-law regime after the phase transition.

Phase transitions in compute-optimal validation losses occur across different task sizes (see Technical Appendix Figure 8). For addition, phase transitions are observed across task sizes and similarly within individual training runs (Technical Appendix Figure 9). As the input size increases, the Pareto frontier shifts to the right but maintains the same shape. However, for maximum subarray, the phase transition only appears at task sizes greater than 16.

4 Feature Learning before Phase Transitions

In order to better understand the observed phase transitions, we investigate the emergence of human-interpretable features during learning. We focus on features corresponding to intermediate outputs of standard algorithms used to perform the tasks. We use linear probing to identify whether the model learned these features.

Feature Probing Methodology

For each algorithm-specific feature, we train separate linear probes across each token position and each layer. Probes are trained on the residual streams after each layer (See Technical Appendix B). Each probe is trained with 10,000 examples which had been held-out from the original model training set.

For each task, we aim to identify the smallest compute budget at which a feature emerges. We select a single model size to study for this task; models of this size are trained for different compute budgets.³ We train separate linear probes for each (model, token position, layer) triple. For each (model, token position) pair, we select the probe that achieves the lowest training loss across layers. We report the test performance of the selected probe for each (model, token position) pair.

We establish random baselines by applying the same probing methodology to models initialized with random weights. Test loss is estimated on a separate test set of 1,000 unseen examples.

²These training runs correspond to compute-optimal hyperparameter settings.

³The model size is chosen so that it is nearly optimal across compute budgets. A fixed model size is chosen in order to make feature metrics easier to compare for models across training runs.

Quiet Features

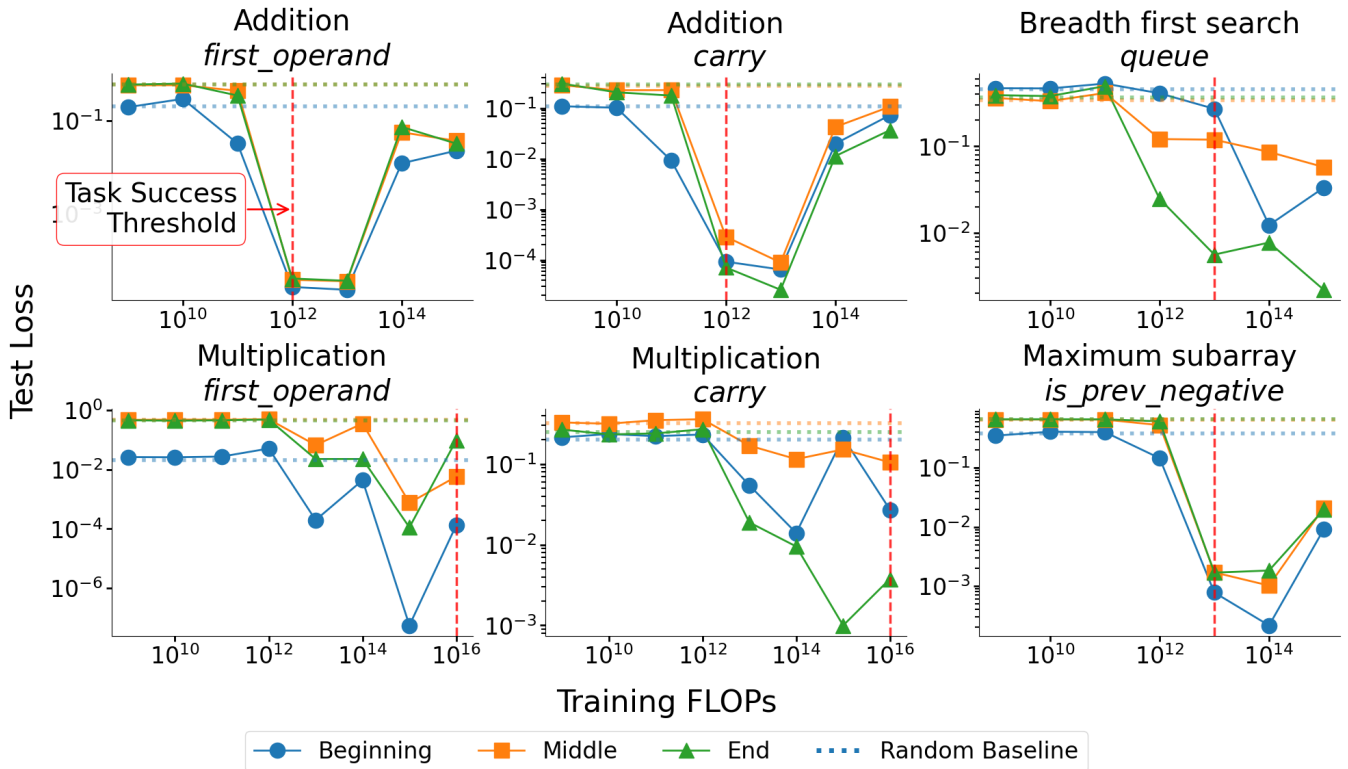


Figure 3: Models learn quiet features before the phase transition. The loss is averaged over the first third of token positions (Beginning), second third (Middle), and last third (End). The red vertical line indicates the task success threshold, which is the smallest compute budget at which the task loss starts to decrease (see Technical Appendix Figure 6). Horizontal dotted lines represent random baselines.

Intermediate Task Features

We describe the features investigated for each of the tasks which exhibit phase transitions in their loss. These features are intermediate values computed in standard algorithms for the tasks.

Addition & Multiplication. For n -bit binary addition, we probe for the following at each token z_i : *first_operand*, which is input bit x_{i+1} (required to compute z_{i+1}); and *carry* c_i , the carry bit used to compute z_{i+1} . Carry c_0 for z_1 is not considered since the first carry is always zero. For multiplication, we check whether the model learns carries generated when adding the last partial product to the sum of the previous $n - 1$ partial products.

Breadth/Depth First Search. For breadth first search, we probe at each token v_{t_i} for the following: *queue*, which is the set of vertices on the queue (in the standard search algorithms) after we have explored vertex v_{t_i} ; and *adjacency_list*, which is the set of vertices adjacent to v_{t_i} .

Maximum Subarray. For the maximum subarray problem, we probe at each token k_i (before the = token) for: *is_prev_negative*, which represents whether k_{i-1} is negative;

and *max_ending_here*, which is the maximum sum of the contiguous subarray ending at k_i . (Refer to Kadane (2023) for the standard algorithm.)

Activity Selection. For the activity selection problem, we probe at each token f_i for *start_time*, which is the corresponding start time s_i . Since the model has to output $s_i f_i$ in order, it must know which start times correspond to which finish times. (Refer to Kleinberg and Tardos (2005) for the standard algorithm.)

Feature Probing Results

The model learns algorithmic features before, during and after the phase transition. We call features learned prior to the phase transition *quiet features*, as they occur during the slow phase where loss is stagnant or slowly decreasing. Features learned in the fast phase (during and after the phase transition) are *loud features* as the task loss decreases rapidly. Figure 3 shows the trajectory of the probing loss for *quiet features*. For addition, multiplication, and maximum subarray, the model learns features for early token positions prior to the phase transition. However, for breadth-first search, later token positions are learned first.

Quiet Features within Single Training Runs

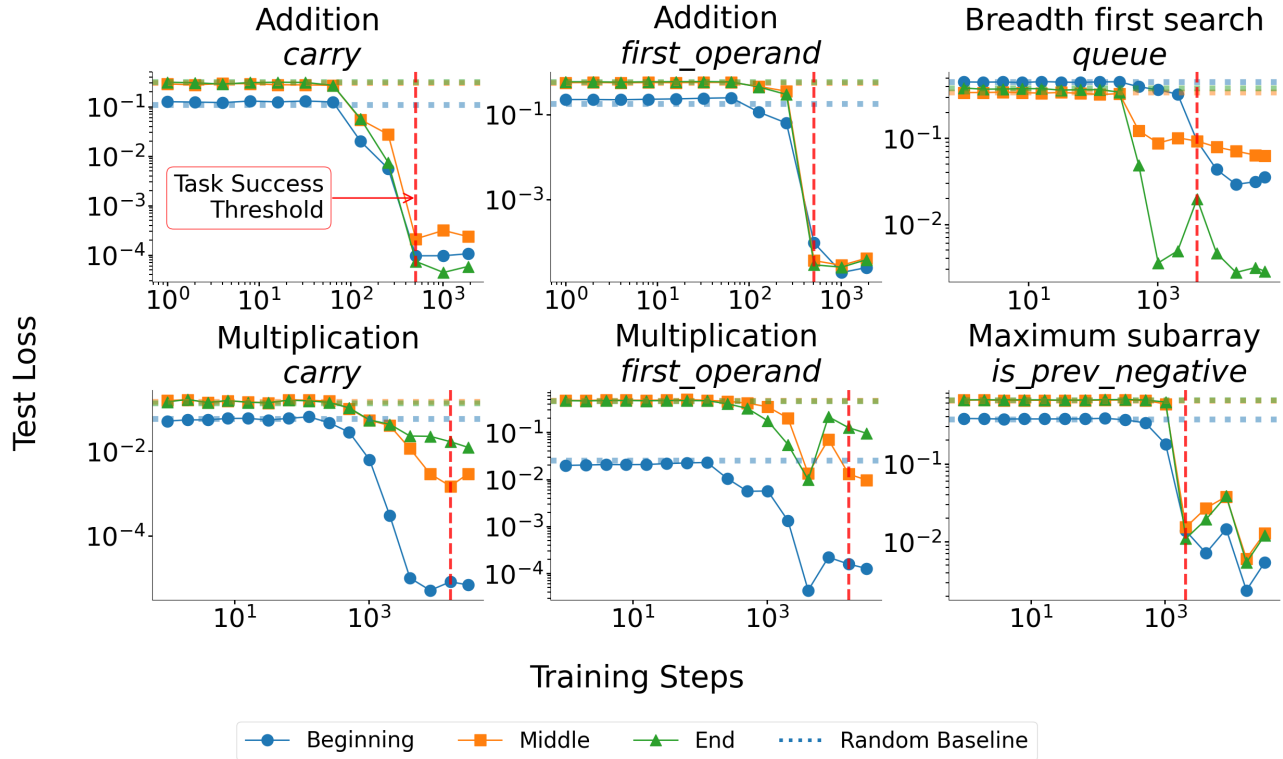


Figure 4: Models learn quiet features before the phase transition within single training runs. The loss is averaged over the first third of token positions (Beginning), second third (Middle), and last third (End). Plots show compute-optimal training runs for the smallest compute budget where test accuracy is 100%. The red vertical line indicates the task success threshold, which is the training step at which the task loss starts to decrease. Horizontal dotted lines represent random baselines.

These results apply across distinct, compute-optimal training runs (for a fixed model size). Figure 4 shows that quiet features also emerge during individual training runs. Features for early token positions are also generally learned first in this case.

Figure 5 shows models learn *loud features* in the fast phase (during and after the phase transition).

A surprising finding is the U-shape of feature learning curves in Figure 3, indicating that the probing loss increases for many quiet features after the phase transition. This may indicate that the models are learning alternative representations in the highest compute budget regimes, though the probing loss remains below the random baseline.

Are quiet features causal?

For a given task, we ablate a *quiet feature* from the residual stream at each position, using the feature probes (one probe per position) identified the previous section. We restrict our analyses to binary features. By comparing to ablations of random features, we can evaluate whether a quiet feature is causally responsible for task performance.

We ablate a feature by removing its direction from the residual stream. A linear feature probe $w^\top x^* + b$ outputs 0 (assigns 0.5 probability to each label) when it detects no

information from the ablated residual stream at that layer. Letting x be the residual stream at a desired layer, we perform the following optimization:

$$\begin{aligned} \operatorname{argmin}_{x^*} \quad & \|x - x^*\|^2 \\ \text{subject to} \quad & w^\top x^* + b = 0. \end{aligned}$$

Solving this yields $x^* = x - \frac{w^\top x + b}{\|w\|^2} w$. The residual stream activation at the linear probe’s layer is replaced with x^* .

Ablation results are shown in Table 1. Quiet feature ablations are compared to ablating a random direction. When we ablate quiet features, we observe test accuracy generally degrades more than ablating a random direction, indicating a causal role for quiet features. Similar results are seen for test loss, as shown in Technical Appendix Table 6. However, for *first_operand* at input size 16, we do not see any significant change after ablating the feature compared to random in addition or multiplication. At larger input sizes, ablating *first_operand* for addition leads to significant test loss degradation compared to random ablation.

5 Discussion

Our findings show that, across different algorithmic tasks, there is often a long phase of training with little appar-

Loud Features

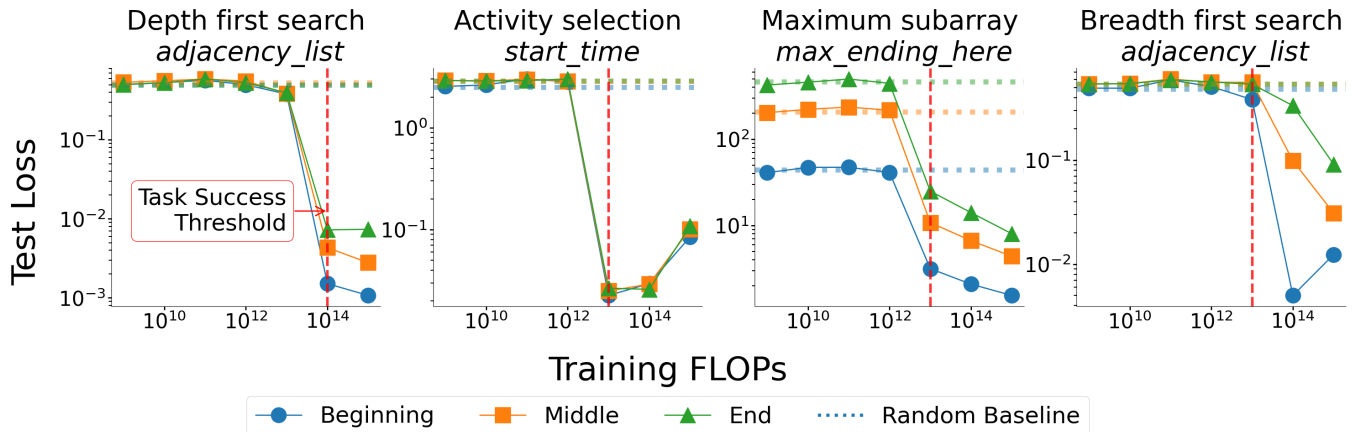


Figure 5: Models learn different set of features (loud features) at or after the phase transition. The loss is averaged over the first third of token positions (Beginning), second third (Middle), and last third (End). The red vertical line indicates the task success threshold, which is the smallest compute budget at which the task loss starts to decrease (see Technical Appendix Figure 6). Cross-entropy loss is used for training probes for *first_operand*, *adjacency_list*. The probing loss is mean squared error for *start_time* and *max_ending_here*, since these are continuous features.

Task	Feature	Feature Ablation Δ Accuracy (%)
Addition (16)	<i>carry</i>	-41.2*
Addition (32)	<i>carry</i>	-50.4*
Addition (64)	<i>carry</i>	-75.1*
Addition (16)	<i>first_operand</i>	0.00
Addition (32)	<i>first_operand</i>	-92.7*
Addition (64)	<i>first_operand</i>	-6.40*
Multiplication (16)	<i>carry</i>	-20.3*
Multiplication (16)	<i>first_operand</i>	-0.05
Maximum Subarray (64)	<i>is_prev_negative</i>	-4.14*
Breadth first search (11)	<i>queue</i>	-43.6*

Table 1: Average difference in test accuracy after ablating a quiet feature compared to ablating a random direction (random ablation). Ablating quiet features degrades test accuracy more than random ablation. For random ablation, we estimate test accuracy over 32 trials. * indicates $p < 0.001$ using bootstrapping. For complete accuracy / loss values see Technical Appendix Table 6

ent improvement in next-token prediction loss. Despite this plateau, we observe that essential internal features (e.g., carry bits in binary addition, adjacency in breadth first search) emerge during these periods. These quiet features emerge prior to any substantial improvement in task performance. Ablation experiments confirm that these features are causally important to solving the tasks, suggesting that models can accumulate partial competence that does not immediately translate into lower loss.

One reason for this quiet period may be the all-or-nothing nature of these tasks: obtaining just some of the required subroutines (e.g., some correct carry bits) does not prevent errors on next token prediction. Consequently, any reduction in loss is small until all sub-features are aligned. In over-parameterized models, there is sufficient capacity to learn these subroutines in the background, allowing partial solutions to remain in the representations until they can be combined into a correct overall procedure.

These findings have practical and conceptual implications. For practitioners, they highlight the risk of judging model capabilities based solely on loss curves. Probe or circuit-based diagnostics could provide earlier warnings that a model is nearing a capability threshold. Conceptually, they raise questions about whether similar quiet phases exist in more complex natural-language settings. They also underscore the need for theoretical frameworks that explain why models accumulate latent subroutines before they begin to pay off in observable metrics.

6 Conclusion

We observe Transformer-based models for algorithmic tasks encode important intermediate computations well before they show significant gains in next-token prediction. This *quiet period* exposes a gap between internal representation learning and external task performance, indicating that sub-features may lie dormant until the final pieces align. We hope these insights motivate new methods for probing and monitoring internal learning dynamics – particularly in larger, more complex models – where hidden phases of progress may likewise precede sudden improvements in capability.

References

- Barak, B.; Edelman, B. L.; Goel, S.; Kakade, S. M.; Malach, E.; and Zhang, C. 2022. Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.
- Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; Goyal, A.; Hartshorn, A.; Yang, A.; Mitra, A.; Srivankumar, A.; Korenev, A.; Hinsvark, A.; Rao, A.; Zhang, A.; Rodriguez, A.; Gregerson, A.; Spataru, A.; Rozière, B.; Biron, B.; Tang, B.; Chern, B.; Caucheteux, C.; Nayak, C.; Bi, C.; Marra, C.; McConnell, C.; Keller, C.; Touret, C.; Wu, C.; Wong, C.; Ferrer, C. C.; Nikolaidis, C.; Allonsius, D.; Song, D.; Pintz, D.; Livshits, D.; Esiobu, D.; Choudhary, D.; Mahajan, D.; Garcia-Olano, D.; Perino, D.; Hupkes, D.; Lakomkin, E.; AlBadawy, E.; Lobanova, E.; Dinan, E.; Smith, E. M.; Radenovic, F.; Zhang, F.; Synnaeve, G.; Lee, G.; Anderson, G. L.; Nail, G.; Mialon, G.; Pang, G.; Cucurell, G.; Nguyen, H.; Korevaar, H.; Xu, H.; Touvron, H.; Zarov, I.; Ibarra, I. A.; Kloumann, I. M.; Misra, I.; Evtimov, I.; Copet, J.; Lee, J.; Geffert, J.; Vranes, J.; Park, J.; Mahadeokar, J.; Shah, J.; van der Linde, J.; Billock, J.; Hong, J.; Lee, J.; Fu, J.; Chi, J.; Huang, J.; Liu, J.; Wang, J.; Yu, J.; Bitton, J.; Spisak, J.; Park, J.; Rocca, J.; Johnston, J.; Saxe, J.; Jia, J.; Alwala, K. V.; Upasani, K.; Plawiak, K.; Li, K.; Heafield, K.; Stone, K.; and et al. 2024. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783.
- Edelman, E.; Tsilivis, N.; Edelman, B. L.; Malach, E.; and Goel, S. 2024. The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Ganguli, D.; Hernandez, D.; Lovitt, L.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; DasSarma, N.; Drain, D.; Elhage, N.; Showk, S. E.; Fort, S.; Hatfield-Dodds, Z.; Henighan, T.; Johnston, S.; Jones, A.; Joseph, N.; Kernian, J.; Kravec, S.; Mann, B.; Nanda, N.; Ndousse, K.; Olsson, C.; Amodei, D.; Brown, T. B.; Kaplan, J.; McCandlish, S.; Olah, C.; Amodei, D.; and Clark, J. 2022. Predictability and Surprise in Large Generative Models. In *FACCT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, 1747–1764. ACM.
- Garg, S.; Tsipras, D.; Liang, P.; and Valiant, G. 2022. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Gu, A.; and Dao, T. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *First Conference on Language Modeling*.
- Henighan, T.; Kaplan, J.; Katz, M.; Chen, M.; Hesse, C.; Jackson, J.; Jun, H.; Brown, T. B.; Dhariwal, P.; Gray, S.; Hallacy, C.; Mann, B.; Radford, A.; Ramesh, A.; Ryder, N.; Ziegler, D. M.; Schulman, J.; Amodei, D.; and McCandlish, S. 2020. Scaling Laws for Autoregressive Generative Modeling. arXiv:2010.14701.
- Hestness, J.; Narang, S.; Ardalani, N.; Damos, G.; Jun, H.; Kianinejad, H.; Patwary, M. M. A.; Yang, Y.; and Zhou, Y. 2017. Deep Learning Scaling is Predictable, Empirically. arXiv:1712.00409.
- Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; de Las Casas, D.; Hendricks, L. A.; Welbl, J.; Clark, A.; Hennigan, T.; Noland, E.; Millican, K.; van den Driessche, G.; Damoc, B.; Guy, A.; Osindero, S.; Simonyan, K.; Elsen, E.; Rae, J. W.; Vinyals, O.; and Sifre, L. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556.
- Hutter, M. 2021. Learning Curve Theory. arXiv:2102.04074.
- Kadane, J. B. 2023. Two Kadane Algorithms for the Maximum Sum Subarray Problem. *Algorithms*, 16(11): 519.
- Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361.
- Kleinberg, J.; and Tardos, E. 2005. *Algorithm Design*. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN 0321295358.
- Lee, N.; Sreenivasan, K.; Lee, J. D.; Lee, K.; and Papailiopoulos, D. 2024. Teaching Arithmetic to Small Transformers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.
- Mallinar, N.; Beaglehole, D.; Zhu, L.; Radhakrishnan, A.; Pandit, P.; and Belkin, M. 2025. Emergence in non-neural models: grokking modular arithmetic via average gradient outer product. arXiv:2407.20199.

Michaud, E. J.; Liu, Z.; Girit, U.; and Tegmark, M. 2023. The Quantization Model of Neural Scaling. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nanda, N.; Chan, L.; Lieberum, T.; Smith, J.; and Steinhardt, J. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Johnston, S.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2022. In-context Learning and Induction Heads. <https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html>.

OpenAI. 2023. GPT-4 Technical Report. *CoRR*, abs/2303.08774.

Power, A.; Burda, Y.; Edwards, H.; Babuschkin, I.; and Misra, V. 2022. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. arXiv:2201.02177.

Ruan, Y.; Maddison, C. J.; and Hashimoto, T. 2024. Observational Scaling Laws and the Predictability of Language Model Performance. arXiv:2405.10938.

Schaeffer, R.; Miranda, B.; and Koyejo, S. 2023. Are Emergent Abilities of Large Language Models a Mirage? In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Varma, V.; Shah, R.; Kenton, Z.; Kramár, J.; and Kumar, R. 2023. Explaining grokking through circuit efficiency. arXiv:2309.02390.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*. Survey Certification.