Learning Modular Exponentiation with Transformers

David Demitri Africa* **Sara M. Kapoor*** **Theo Simon Sorg***
Challenger Mishra
Department of Computer Science and Technology
University of Cambridge

Abstract

Modular exponentiation ($a^b \equiv d \bmod c$) is crucial to number theory and cryptography, yet remains largely unexplored from a mechanistic interpretability standpoint. We train compact 4-layer encoder–decoder Transformers to predict d and analyze how they come to solve the task. We compare principled sampling schemes for (a, b, c, d), probe the learned token embeddings, and use causal interventions (activation patching) to localize the computation inside the network. Sampling a and b log-uniformly (reciprocal sampling) removes severe output imbalance and yields large accuracy gains, with abrupt, synchronized jumps in accuracy that simultaneously cover families of related moduli (e.g., multiples of 23). Causal analysis shows that, on instances without reduction ($c > a^b$), a small circuit consisting only of final-layer attention heads reproduces full-model behavior, indicating functional specialization. These results suggest that Transformers can internalize modular arithmetic via compact, specialized circuits, and that data distribution strongly shapes both learning dynamics and generalization.

1 Introduction

Modular exponentiation is fundamental in cryptographic algorithms such as RSA [Rivest et al., 1978] and Diffie-Hellman key exchange [Diffie and Hellman, 1976]. Despite its mathematical simplicity, modular exponentiation presents significant challenges for machine learning models, given the non-linear and cyclic nature of the operation. Prior work by Charton [2024] explored how transformers can learn arithmetic functions such as the greatest common divisor (GCD), showing that transformers implicitly uncover algorithmic structures in arithmetic tasks. Computationally, modular exponentiation is efficiently solvable, as the decision problem "is $a^b \equiv d \pmod{c}$?" lies in P since repeated squaring computes $a^b \bmod c$ with $O(\log b)$ modular multiplications (overall bit-complexity $O(M(n) \log b)$ for n-bit inputs) [Knuth, 1969, Von Zur Gathen and Gerhard, 2003, Menezes et al., 2018].

Given the interaction between exponentiation and modulo reduction, modular exponentiation may present nontrivial number-theoretic patterns not present in simpler tasks. Transformer-based models have not been systematically investigated for modular exponentiation, particularly from a mechanistic interpretability perspective. As such, uncovering novel patterns in modular exponentiation may also have knock-on effects in the study of diophantine geometry and modularity. In general, machine driven mathematical discovery has been a rich area of exploration, with large datasets available for machine learning exploration in conjecture generation [Davies et al., 2021, He et al., 2024, Wang et al., 2025].

In this work, we train transformer models to perform modular exponentiation, design novel sampling methods to capture the statistical characteristics of modular arithmetic, and analyze how these sampling strategies influence model learning dynamics and generalization. We then use mechanistic

*Equal contribution. Correspondence: `{dda28,smk78,tss52}@cam.ac.uk` interpretability methods to understand how models represent number-theoretic information internally. We also observe evidence of grokking as the model learns individual multiples.

2 Related Work

Mechanistic interpretability seeks to uncover computational strategies internalized by neural networks at the level of individual components such as attention heads and weight matrices. Olah et al. [2020] introduced the concept of *circuits*, coherent subgraphs corresponding to meaningful algorithmic operations, with subsequent work by Elhage et al. [2021] and Wang et al. [2022] expanding this framework to dissect transformer internal structures. Recent mechanistic interpretability has been applied specifically to arithmetic reasoning, with Quirke and Barez [2024] discovering dedicated attention heads mirroring human arithmetic algorithms and Stolfo et al. [2023] using causal mediation analysis to reveal interpretable arithmetic pathways. Machine learning approaches to modular arithmetic have focused on simpler operations like modular addition [Gromov, 2023, Saxena et al., 2024], with Gromov [2023] demonstrating neural networks learning modular addition through *grokking*, sudden generalization from memorization to algorithmic understanding initially documented by Power et al. [2022]. Nanda et al. [2023] introduced internal progress measures to identify circuits responsible for emergent arithmetic capability, while Doshi et al. [2024] studied grokking in modular polynomial arithmetic. Our work extends these explainability techniques to the previously unexplored domain of modular exponentiation.

3 Sample Generation and Model Training

Following Charton [2024], we experiment with various sampling methods for modular exponentiation. Modular exponentiation requires sampling integers $a, b, c \in \mathbb{Z}$ and outcome $d \in \mathbb{Z}$ such that

$$a^b \equiv d \mod c \tag{1}$$

We sample a, b, c and sometimes d such that tuples (a, b, c, d) follow a certain underlying distribution, with the maximum integer to sample set to $M = 10^6$. We find that different distributions lead to varying learning dynamics and absolute accuracies, with greater imbalance leading to lower accuracy.

Uniform operands samples $c \in [1, 100]$ and $a, b \in [0, M]$ uniformly, then computes d. This creates severe class imbalance with d heavily skewed toward small values, preventing the model from learning larger outcomes.

Uniform outcomes mitigates this by sampling d uniformly, then rejection sampling (a, b, c) such that (1) holds with constraint $c > d$.

Reciprocal operands samples a, b log-uniformly by sampling $\ln(a), \ln(b) \sim \mathcal{U}(1, \ln(M + 2))$, computing $a = \lfloor e^{\ln(a)} \rfloor, b = \lfloor e^{\ln(b)} \rfloor$, then shifting by 1 to include $a, b = 0$. This yields the discrete probability distribution:

$$\mathcal{P}(n, 0, M) = \begin{cases} 0 & n < 0 \text{ or } n > M \\ \frac{\ln(n+2) - \ln(n+1)}{\ln(M+2)} & 0 \leq n \leq M \end{cases} \tag{2}$$

We detail the proof for this in Appendix A.1. In training, we test four combinations: uniform/reciprocal operands × computed/uniform outcomes. For comparability with Charton [2024] train four-layer encoder-decoder transformers with embedding dimension 256, eight attention heads, batch size 256, and learning rate 10^{-4} with the Adam optimizer [Kingma and Ba, 2014]. Each epoch uses 300,000 generated samples.

Integer representations Since transformers operate on discrete tokens, and the range of integers up to M is too large to use as a vocabulary for the small transformers trained here, we follow Charton [2024] and represent integers using base B digits in the template:

$$\text{V3 } + a_1 \dots a_n + b_1 \dots b_n + c_1 \dots c_n + d_1 \dots d_n$$

For example, $750178^{996884} \equiv 1 \mod 95$ becomes V3 + 750 178 + 996 884 + 95 + 1 in base 1000. This string is constructed using the samples generated during training.

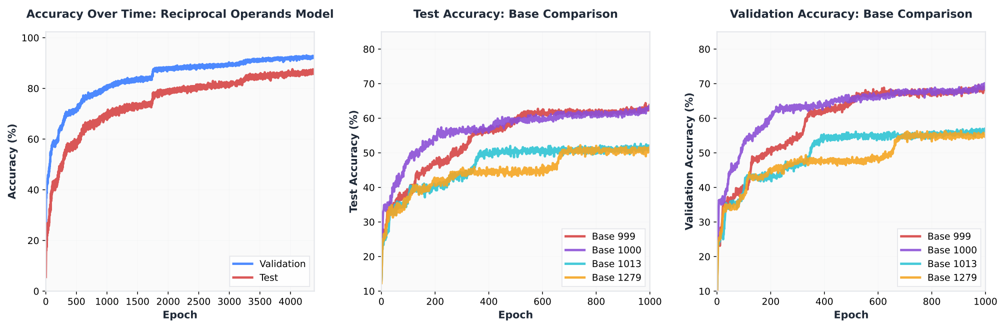


Figure 1: *Left*: Validation and test accuracy over 3000 epochs for the reciprocal operands model. Reciprocal sampling (log-uniform distribution of operands) enables effective learning of modular exponentiation, with validation accuracy reaching ∼84% and test accuracy ∼80%. *Middle*: Test accuracy comparison across four numerical bases over 1000 epochs. Composite bases (999, 1000) substantially outperform prime bases (1013, 1279), with bases 999 and 1000 reaching ∼60% accuracy compared to ∼49% for prime bases. *Right*: Validation accuracy shows the same pattern, confirming that the composite base advantage generalizes across both evaluation sets. Base choice significantly impacts learning dynamics and final performance.

Evaluation We test all four sampling methods for 2500 epochs using base 1000, with validation set (uniform operands, computed outcomes) and test set (uniform operands, uniform outcomes) to assess performance across distributions. We also test four bases on the best-performing reciprocal operands setting to study the impact of base choice.

4 Results

Transformer models successfully learn modular exponentiation, with the best performing model reaching over 80% test accuracy after 3000 epochs (Figure 1).

Performance on modular exponentiation. Reciprocal operands sampling yields dramatically better performance than uniform operands (Table 1), resolving class imbalance without requiring uniform outcomes. In the following, samples are generated using reciprocal operands if not explicitly mentioned otherwise. Prime number bases clearly perform worse, although it is not clear why[2]. However, when comparing the two prime numbers or composite numbers with each other, there is no clear advantage. Both prime numbers and the composite numbers perform equally well, respectively. The subsequent experiments were conducted with base 1000, as bases 1000 and 999 achieve similar accuracy.

	Computed	Uniform
Uniform operands	13.17	28.14
Reciprocal operands	**80.39**	79.16

Table 1: Test accuracy (%) for sampling methods.

Evaluating deterministic predictions. Following Charton [2024], we analyze deterministic mispredictions. The model consistently predicts 19 instead of the correct 91 across all 10,012 samples (512 distinct ones) with target 91. When we control for duplicates and scale up samples with fixed $d = 91$, prediction 1 becomes most common for target 91, with 19 persisting for infrequent a, b, c. We hypothesize that the 1 might serve as a fall-back mechanism for our model, which would be relevant for unseen and rare data.

Learning dynamics analysis. We observe significant performance surges between epochs 1725-1750, with simultaneous accuracy increases from 20% to 100% for multiples of 23 (moduli 23, 46, 69,

[2]We suspect that composite bases like $1000 = 2^3 5^3$ may facilitate learning by aligning with divisibility structure in modular arithmetic, though further investigation is needed.

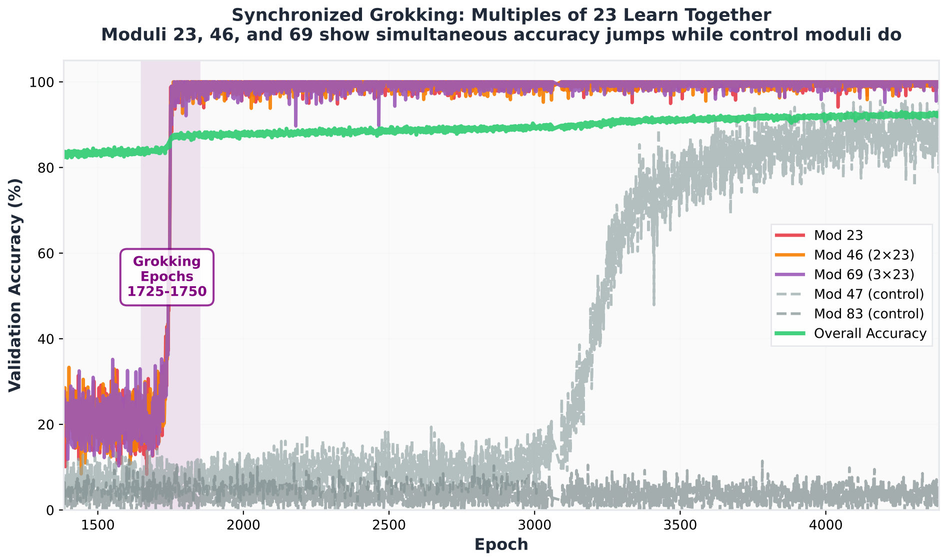


Figure 2: Synchronized grokking for multiples of 23 during epochs 1725–1750 (highlighted region). Moduli 23, 46 (2×23), and 69 (3×23) exhibit simultaneous accuracy jumps from $\sim$20% to near-perfect performance, demonstrating that the transformer discovers and exploits mathematical relationships between related moduli. Control moduli 47 and 83 (not multiples of 23) exhibit a different learning pattern with gradual improvement, while overall accuracy remains high ($\sim$83%) throughout training.

92), as shown in Figure 2. Similar effects occur for multiples of 31, 39, and 47. Further visualizations can be found in Appendix A.3. Small moduli (1, 2, 4, 10, 12, 14, 15, 18) are learned within 100 epochs, while others exhibit stepwise grokking behavior. This moduli-specific learning suggests the model discovers mathematically meaningful functions. Therefore, we explore whether the underlying representations encapsulate relations between integers.

Visualizing the embedding space. We perform PCA on token embeddings (tokens 1-100) before and after grokking to examine numerical patterns. We analyze embeddings by numeric value, lowest prime factor, parity, primality, divisor count, multiplicative order, Euler's totient function $\phi(n)$, primitive roots, residue classes modulo 5, and multiples of specific numbers (23, 31, 39). Before grokking, embeddings form spatially distinct clusters with weak structure for most number-theoretic properties. After grokking, embeddings become more centralized and compressed, though clear clustering by mathematical properties remains limited. The general centralization suggests structural reorganization during learning, but interpretable mathematical organization is not clearly evident. Additional PCA visualizations are provided in Appendix A.2.

Results of activation patching. We use activation patching [Heimersheim and Nanda, 2024] to identify minimal circuits by replacing attention head activations with counterfactual inputs and measuring KL divergence. We find that regular exponentiation (when $c > a^b$) can be performed using only final-layer attention heads, achieving full model accuracy with a substantially smaller circuit, suggesting functional specialization where higher layers encode task-specific transformations. Using 100 prompt-counterfactual pairs, circuit analysis reveals that final-layer attention heads alone achieve full model accuracy, with earlier layers having minimal causal impact (Figure 3). This suggests functional specialization where higher layers encode task-specific transformations.

5 Discussion

Our study extends transformer arithmetic learning to modular exponentiation. Reciprocal operand sampling achieves over 80% test accuracy by resolving output space imbalance, echoing Charton [2024]'s findings that distributional choices significantly impact learning. The learning dynamics

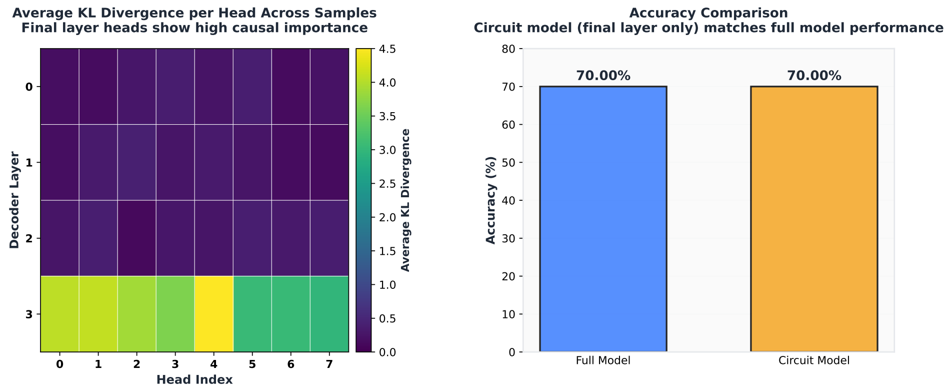


Figure 3: *Left*: KL divergence heatmap showing causal importance of each attention head across the 4-layer decoder. Warmer colors indicate higher KL divergence between clean and patched activations, reflecting greater causal impact on model predictions. Final-layer heads (layer 3) exhibit substantially higher KL divergence (∼3–4.5) compared to earlier layers (∼0.1–0.4), indicating that the circuit for regular exponentiation (when $c > a^b$) is concentrated in the final decoder layer. *Right*: Accuracy comparison between the full model and the minimal circuit consisting only of final-layer attention heads. Both achieve 70% accuracy on regular exponentiation tasks, demonstrating that earlier layers contribute negligibly to this computation and confirming functional specialization in the network architecture.

reveal moduli-specific grokking where accuracy surges coincide with solving sets of related moduli (e.g., multiples of 23), mirroring sieve-like learning in prior GCD work.

PCA analysis shows embedding centralization post-grokking, though clear clustering by number-theoretic properties remains limited. The limited mathematical clustering in embeddings that we do observe suggests transformers may encode modular arithmetic through distributed representations rather than explicit symbolic groupings. The centralization post-grokking indicates structural reorganization, but further work with probing classifiers or feature attribution methods could better characterize what mathematical properties are captured. We also observe deterministic mispredictions (e.g., predicting 19 instead of 91) suggesting fallback mechanisms for rare inputs. Activation patching demonstrates that regular exponentiation uses only final-layer circuits, indicating functional specialization where transformers compartmentalize arithmetic operations.

6 Conclusion

We demonstrate that transformers can learn modular exponentiation with high accuracy using reciprocal sampling strategies. Key findings include stepwise grokking of related moduli, embedding space reorganization, and specialized circuits for arithmetic operations.

Limitations. Our findings are limited to compact transformers; scaling to larger architectures (e.g., 12+ layers, hundreds of attention heads) may reveal different circuit structures or learning dynamics. Future work should investigate whether specialized circuits persist or become distributed in larger models. We focus on synthetic data with moduli up to 100 and operands up to 10^6. Real cryptographic applications use much larger bit-lengths (e.g., 2048-bit RSA). Our findings mainly demonstrate proof-of-concept for mechanistic understanding. Further, as our goal is mechanistic interpretability of transformers specifically, we do not compare against simpler sequence models (RNNs, LSTMs) or explicit algorithmic implementations.

Acknowledgements

We thank the reviewers for helpful feedback on limitations and scope. David Demitri Africa is supported by the Cambridge Trust and the Jardine Foundation.

References

F. Charton. Learning the greatest common divisor: explaining transformer predictions. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=cmcD05NPKa`.

A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi: 10.1109/TIT.1976.1055638.

D. Doshi, B. Zhu, P. Abbeel, et al. Grokking modular polynomials. *arXiv preprint arXiv:2401.09356*, 2024. URL `https://arxiv.org/abs/2401.09356`.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, L. Lovitt, L. Schubert, C. Olah, D. Amodei, S. McCandlish, T. Brown, J. Kaplan, and J. Clark. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL `https://transformer-circuits.pub/2021/framework/index.html`.

A. Gromov. Learning modular addition through grokking dynamics in neural networks. *arXiv preprint arXiv:2301.12345*, 2023.

Y.-H. He, K.-H. Lee, T. Oliver, and A. Pozdnyakov. Murmurations of elliptic curves. *Experimental Mathematics*, pages 1–13, 2024.

S. Heimersheim and N. Nanda. How to use and interpret activation patching, 2024. URL `https://arxiv.org/abs/2404.15255`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. Knuth. Vol. 2: Seminumerical algorithms. *The Art of Computer Programming*, 1969.

A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

N. Nanda, N. Elhage, S. Ganguli, C. Olah, et al. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://arxiv.org/abs/2211.10954`.

C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. URL `https://distill.pub/2020/circuits/zoom-in/`.

A. Power et al. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022. URL `https://arxiv.org/abs/2201.02177`.

P. Quirke and F. Barez. Understanding addition in transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/`. OpenReview.net.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

P. Saxena, J. Li, and R. Chen. Modular arithmetic with transformers: Learning to add and multiply modulo n. In *Proceedings of the 2024 International Conference on Machine Learning*. PMLR, 2024.

A. Stolfo, Y. Belinkov, and M. Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL `https://arxiv.org/abs/2310.07041`.

J. Von Zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge university press, 2003.

K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022. URL `https://arxiv.org/abs/2211.00593`.

Y. Wang, M. Bennani, J. Martens, S. Racanière, S. Blackwell, A. Matthews, S. Nikolov, G. Cao-Labora, D. S. Park, M. Arjovsky, D. Worrall, C. Qin, F. Alet, B. Kozlovskii, N. Tomašev, A. Davies, P. Kohli, T. Buckmaster, B. Georgiev, J. Gómez-Serrano, R. Jiang, and C.-Y. Lai. Discovery of unstable singularities, 2025. URL `https://arxiv.org/abs/2509.14185`.

A Appendix / supplemental material

A.1 Reciprocal operand distribution

We would like to compute the probability distribution $G(n, 0, M)$ for given $n \in \mathbb{Z}$ and M using the reciprocal distribution $F(x, 1, M+1)$. Note that the flooring simply implies squashing the probability mass of all values $n \leq x < n+1$. Given the cumulative probability distribution $F(x, a, b)$ and the probability density function $f(x, a, b)$ of a reciprocal distribution defined on all reals, we have a distribution on the integers

$$H(n, a, b) = \int_{x \in [n, n+1]} f(x, a, b)\, dx = F(n+1, a, b) - F(n, a, b)$$

This can be case-split into

$$H(n, a, b) = \begin{cases} 0 & n < a \text{ or } n \geq b \\ \frac{\ln(n+1) - \ln(n)}{\ln(b) - \ln(a)} & a \leq n \leq b-1 \end{cases}$$

With $Y := X - 1$, $X \sim H$ as our shifted integer for our final distribution P, we substitute and simplify bounds and get for $Y = n'$

$$P(Y = n', a, b) = H(n'+1, a, b) = \begin{cases} 0 & n' < a-1 \text{ or } n' \geq b-1 \\ \frac{\ln(n'+2) - \ln(n'+1)}{\ln(b) - \ln(a)} & a-1 \leq n' \leq b-2 \end{cases}$$

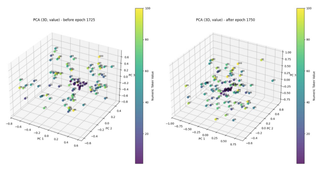
Note that this still relies on the old bounds. To get the formulation used in the main part of the paper, set $a = 1, b = M + 2$.

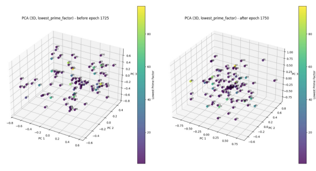
A.2 Additional PCA Embeddings Visualizations

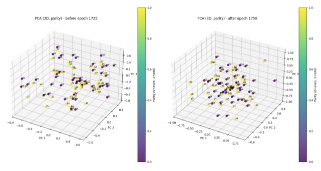
We present additional visualizations of the PCA embeddings presented in 4.4. These are 3D representations of the same 9 number-theoretic metrics, and in addition a visualization by the multiples of 23 for which we observed significant performance increases, outlined in 4.3. As with the other metrics, performing PCA on multiples did not display any notable clusterings before grokking, with a centralization of embeddings emerging post-grokking.

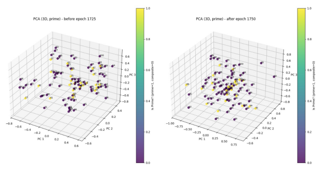
A.3 Additional moduli

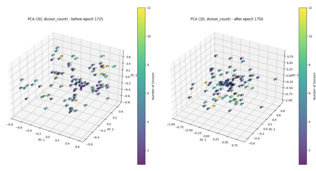
We present some additional charts showing the learning dynamics of various moduli. The second to last number separated by _ encodes the modulus.

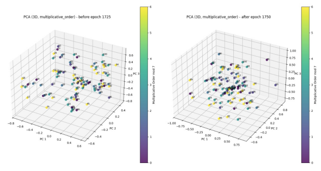

(a) Value (3D)

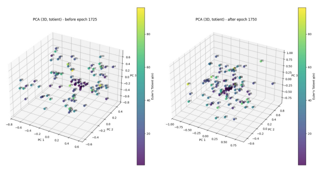

(b) Lowest Prime Factor (3D)

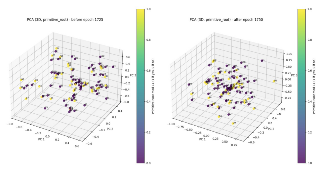

(c) Parity (3D)

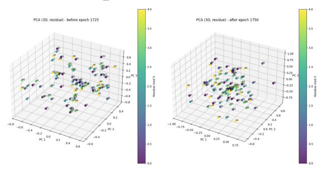

(d) Prime (3D)


(e) Divisor Count (3D)


(f) Multiplicative Order (3D)


(g) Euler’s Totient $\phi(n)$ (3D)


(h) Primitive Root (3D)


(i) Residue mod 5 (3D)

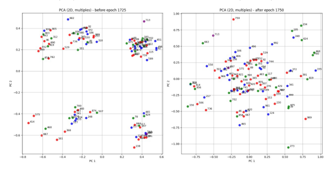

(j) Multiples (2D)

Figure 4: PCA 3D projections of token embeddings, colored by number-theoretic properties, before and after grokking. Bottom row shows multiples in 2D.

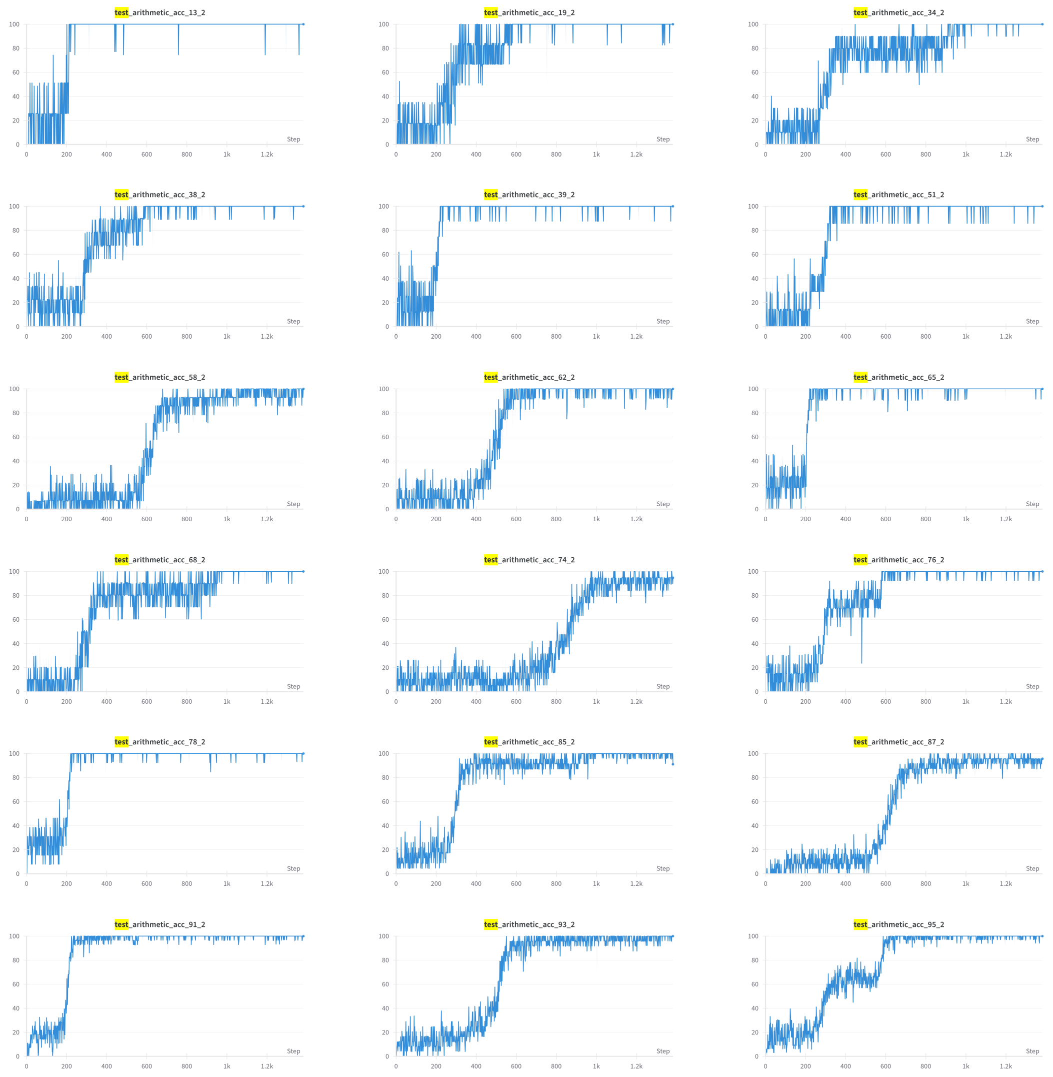


Figure 5: Test plots from training runs for 18 different moduli.