OPERAND ACCESS AND STATE DYNAMICS IN LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped Transformers increase computational depth by repeatedly applying shared parameters across recurrent steps, yet it remains unclear how their internal representations support multi-step computation and extrapolation beyond the training horizon. We study modular product learning in looped Transformers, including progressively masked models and Fixed-Point Reasoning Models, using mechanistic interpretability to examine how operand access shapes their computation and extrapolation. We find that the way operands are exposed across recurrent steps, rather than recurrent depth alone, determines how well models extrapolate. Progressive operand exposure extends the compute horizon for commutative composition but fails for non-abelian composition, while among models that learn the task, sequential exposure generalizes farther beyond the training range than causal access. In one successful model, accuracy decreases from 100.0% at sequence length 8 to 10.0% at length 24, with failure emerging gradually through declining readout margins rather than instability or drift in the recurrent state. Representation analyses reveal distinct patterns of information encoding across successful models, suggesting that task success and length generalization alone do not uniquely determine the internal representation of a solution. Together, these results show that operand-access constraints shape recurrent extrapolation without uniquely determining how the underlying computation is represented.

1 INTRODUCTION

Universal Transformers introduced iterative computation into the Transformer architecture by repeatedly applying a shared transition to evolving latent representations (Dehghani et al., 2019). Looped Transformers extend this by repeatedly applying a shared block across iterations, decoupling computation steps from learned parameters (Giannou et al., 2023) and allowing the effectiveness of a learned update beyond the training regime to be tested directly.

Modular arithmetic provides a tractable setting for studying how such computations are organized. Transformers can learn structured representations of modular operations, including Fourier-based clock circuits for modular addition (Nanda et al., 2023), while recent work extends these settings to multi-operand and compositional tasks with intermediate variables and longer dependencies (Exoo et al., 2026). Multi-operand modular computation therefore provides a setting in which both final performance and the evolution of intermediate computations can be examined across successive steps.

Recent work demonstrates that looped and weight-tied Transformers can support extended computation in language modeling, adaptive reasoning, and algorithmic tasks (Zhu et al., 2025; Yang et al., 2024; Geiping et al., 2025; McLeish et al., 2024). At the same time, repeated application of shared weights can produce distribution-specific fixed points, oversmoothing, and degraded length generalization. To address these issues, deep supervision and adaptive halting have been proposed (Elinas & Bonilla, 2022; Movahedi et al., 2026). However, it remains unclear how successful looped Transformers organize and propagate task-relevant computation across recurrent steps, and which properties of this computation determine whether it remains valid beyond the training horizon.

In this work, we study the mechanisms underlying iterative algebraic computation in looped Transformers, focusing on modular product learning and extending the analysis to non-abelian groups. We make five contributions. (1) We characterize extrapolation in looped Transformers, showing that models can learn reusable iterative updates that extend computation beyond the training

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

horizon. (2) We establish the role of operand-access patterns in governing this extrapolation. (3) We identify the mechanism of extrapolation failure, showing that performance degrades gradually through declining readout margins rather than instability or drift in the recurrent state. (4) We demonstrate that successful solutions are representationally non-unique. (5) We provide a causal neuron-level analysis of iterative modular computation, identifying a small, layer-concentrated set of Fisher-selected MLP neurons whose activation causally influences modular-multiplication predictions. Together, these contributions characterize iterative computation in looped Transformers from the perspectives of extrapolation, information access, failure modes, representation, and causal mechanisms.

2 Related work

2.1 Recurrent and weight-shared transformer architectures

Recurrent and weight-shared Transformer architectures increase computational depth without proportionally increasing parameter count. Universal Transformers repeatedly apply a shared transition to update the hidden state across computation steps (Dehghani et al., 2019), while Looped Transformers make repeated application of a shared block an explicit computational axis (Giannou et al., 2023), decoupling computation depth from parameter count and letting the evolution of latent representations under repeated computation be studied.

Repeated computation introduces optimization challenges, including vanishing gradients and unstable activations (Fu et al., 2026). Step supervision provides intermediate training signals across recurrent steps (Fan et al., 2025), while Fixed-Point Reasoning Models (FPRM) combine step supervision with adaptive halting based on hidden-state stability (Movahedi et al., 2026). We therefore evaluate state stability alongside task accuracy to distinguish convergence from a correct computation.

2.2 Large-scale looped transformers and benchmark tasks

Recent work has demonstrated the scalability of recurrent and weight-tied Transformers to language modeling, reasoning, and algorithmic tasks. Large recurrent-depth models such as Huginn and Ouro show that repeated application of shared blocks can provide additional test-time computation at substantially lower parameter counts than fixed-depth alternatives (Geiping et al., 2025; Zhu et al., 2025): Huginn reaches 3.5B parameters trained on 800B tokens, while Ouro's 1.4B and 2.6B variants, pretrained on 7.7 trillion tokens, match dense models up to 4B and 8B parameters respectively. Mixture-of-Recursions and related architectures further introduce adaptive or token-dependent computation depth, evaluated across models from 135M to 1.7B parameters (Bae et al., 2025), and earlier work demonstrated competitive performance on synthetic in-context learning and algorithmic tasks including addition and sorting (Yang et al., 2024; McLeish et al., 2024). These results establish that recurrent computation can be useful at scale, but primarily evaluate end-task performance rather than internal computation. Repeated application can also produce optimization and representation-level limitations, including oversmoothing, unstable dynamics, and degraded length generalization (Yang et al., 2024; Ruiz & Gu, 2025; Chehreghani, 2026), which benchmark accuracy alone would not reveal.

2.3 Mechanistic interpretability of modular arithmetic

Modular arithmetic provides a tractable setting for mechanistic analysis because learned computations can often be related directly to the algebraic structure of the task. In modular addition, Transformers encode residues using Fourier features and implement addition through phase rotations in interpretable clock circuits (Nanda et al., 2023). Subsequent work identified alternative, non-Fourier circuits, showing the same task can admit qualitatively different internal solutions (Zhong et al., 2023), while related work on modular multiplication finds Transformers organizing representations around discrete-logarithm structure (Nguyen, 2026), and mechanistic studies of Grokking connect the emergence of structured representations with generalization (Liu et al., 2022; Power et al., 2022).

Extending these analyses to multiple operands introduces the problem of composing intermediate computations across steps. Fourier analyses of multi-input modular addition reveal structured representations but largely within a single computational layer (Li et al., 2025), while recent work identifies mechanisms for routing intermediate variables between successive operations (Exoo et al.,

2026). These studies establish tools for analyzing structured arithmetic circuits, but leave open how such representations are maintained and transformed across repeated computational steps.

2.4 Mechanistic interpretability of looped transformers

Mechanistic studies of recurrent Transformers have examined both state convergence and the role of recurrent representations in computation. Universal Transformers use learned halting to terminate recurrent computation (Dehghani et al., 2019), while Deep Equilibrium Models explicitly characterize representations as fixed points of weight-tied transformations (Bai et al., 2019). Recent work on recurrent-depth language models examines convergence toward steady states and shows recurrent trajectories can be path-independent despite different dynamical behaviors (Geiping et al., 2025), and complementary analysis finds that layers within a recurrent block can approach distinct fixed points, producing a stable limit cycle accompanied by stabilizing attention maps (Blayney et al., 2026).

A related line of work treats recurrent states as working memory for an iterative algorithm. Constructive analyses show that looped Transformers can update encoded instructions and memory across computational steps (Giannou et al., 2023), while linear-probe analyses of trained models show that regression-related statistics become increasingly decodable across recurrent iterations (Yang et al., 2024). Theoretical work likewise connects looped computation to iterative optimization algorithms, and recent studies relate recurrent depth to latent algorithms for arithmetic and reasoning (Gatmiry et al., 2024; Saunshi et al., 2025), together indicating that recurrence can organize representations into algorithmic stages. However, how task-relevant information is accessed, propagated, and transformed across recurrent steps in trained looped Transformers, and how these dynamics determine extrapolation beyond the training horizon, remains poorly understood.

3 Methodology

To study how operand access and recurrent computation shape modular product learning in looped Transformers, we first consider modular multiplication over $\mathbb{Z}_{113}$ and compare progressive, sequential, and causal operand-access schemes. We then extend the analysis to non-abelian group composition, where operand order is essential. Finally, we analyze the resulting representations and localize computational circuits using linear probes, logit lenses, algebraic representation analysis, Fisher-based neuron localization, and causal activation patching.

3.1 Looped Transformer and Operand Access

For modular multiplication, each input consists of factors $x_1, \ldots, x_k \in 1, \ldots, 112$ followed by an equals token. The target at position j is the prefix product $y_j = \prod_{i=1}^{j} x_i \pmod{113}$.

We use a looped Transformer consisting of a single pre-LayerNorm Transformer block with parameters shared across recurrent steps. The residual width is 128, self-attention uses four heads, and the ReLU MLP has width 512. Positional embeddings are omitted, and the original input embeddings are reinjected at each recurrent step. Let $h^{(\ell)}$ denote the residual stream after recurrent step ℓ and $x^{(0)}$ the original embedded input. The recurrent update is $h^{(\ell+1)} = F_\theta(h^{(\ell)} + x^{(0)}; M_\ell)$, where F_θ is the shared Transformer block and M_ℓ is the step-dependent attention mask. A shared linear unembedding maps the representation at the equals position to output logits.

The factor tokens retain the standard causal mask in all conditions. The access schemes differ only in which factor tokens the equals position can attend to. Under *progressive* access, the equals position attends to x_1, x_2, and itself at the first recurrent step, and to $x_{\ell+2}$ and itself at subsequent step ℓ. Thus, one additional operand is introduced at each step after the initial two, requiring $k-1$ recurrent steps and producing targets $y_2, \ldots, y_k$. Under *sequential* access, the equals position attends only to $x_{\ell+1}$ and itself at step ℓ, exposing one new factor per recurrent step. These models use k recurrent steps and predict $y_1, \ldots, y_k$. Under *causal* access, the equals position attends to all preceding factors and itself at every recurrent step, providing a full-access control condition.

For representation analyses, we additionally use a *seeded-sequential* condition. The representation of x_1 is written into the recurrent state before the first shared-block application. Each subsequent recurrent step then introduces one additional factor under sequential-style masking. These models use

$k-1$ recurrent steps and predict $y_2, \ldots, y_k$. Seeded-sequential models are used only for representation and terminal-stability analyses and are not part of the primary progressive/sequential/causal comparison. Each productive recurrent step is supervised with cross-entropy against its corresponding prefix-product target.

3.2 Recurrent Updates and Terminal-Stability Interventions

We test whether extrapolation depends on information being carried between recurrent applications. In a *carry-reset* intervention, the recurrent state is reset to zero before each application of the shared block, $h^{(\ell+1)} = F_\theta(x^{(0)}; M_\ell)$. The input embeddings and attention masks are unchanged, isolating the contribution of information carried across recurrent steps.

We additionally compare the standard recurrent update with an under-relaxed Picard-style update, $h^{(\ell+1)} = (1-\alpha)h^{(\ell)} + \alpha F_\theta(h^{(\ell)} + x^{(0)}; M_\ell)$, with $\alpha = 0.5$. The standard update corresponds to $\alpha = 1$. We evaluate both updates under five operand-access conditions: one operand, two operands, four operands, all-seen operands, and all operands, using matched architectures and training procedures.

To test terminal stability, we continue computation after all required operands have been introduced. For progressive modular multiplication, the final mask is held fixed after the $k = 16$ computation, so subsequent recurrent steps introduce no new factor. For seeded-sequential S_4 composition, the corresponding intervention begins after the $k = 8$ computation. These fixed-mask interventions test whether the terminal computation remains stable when the operand-access pattern no longer changes.

The Fixed-Point Reasoning Model (FPRM) provides a separate adaptive-computation setting. It is trained on the same modular multiplication task but learns when to halt rather than using a predetermined number of recurrent steps. Final checkpoints are evaluated for up to 48 adaptive iterations on length-32 sequences, allowing computation to continue beyond the training horizon.

3.3 Non-Abelian Group Composition

We extend the modular-product experiments to non-abelian group composition using A_4, D_6, S_4, and D_{12}. These experiments use a separate NoPE looped Transformer with residual width 256, four attention heads, and an MLP width of 1024. The recurrent block is shared across steps, and operand embeddings are reinjected at every step.

We evaluate the same progressive, sequential, and causal access schemes. Sequential and causal models use k recurrent steps, whereas progressive models use $k-1$ recurrent steps. The target at each step is the ordered group product $y_j = x_1 x_2 \cdots x_j$. Because these groups are non-commutative, operand order is essential. In particular, the first progressive step exposes x_1 and x_2 simultaneously without positional information, so their order is unavailable at that step. This tests whether progressive operand exposure remains effective when composition is order-sensitive.

3.4 Representation Analysis

We analyze recurrent representations at the equals position to characterize when and how intermediate products become encoded. First, at each recurrent step, we train a linear probe consisting of a single linear layer mapping the 128-dimensional residual representation to the corresponding prefix-product class. Held-out activations are split 80/20 for probe training and evaluation. Probe accuracy across recurrent steps measures the emergence of linearly decodable intermediate products.

Second, we apply a logit lens to intermediate MLP outputs. For each neuron, its output-projection vector is mapped through the shared unembedding matrix to obtain output logits. We record the highest-probability output token and its confidence at each recurrent step, allowing product information to be tracked before the final computation.

Third, we analyze whether representations reflect the algebraic structure of the underlying task. For $\mathbb{Z}_{113}^*$, we use the primitive root $g = 3$ and reorder residue embeddings according to their discrete logarithms. We then compute one-dimensional Fourier spectra of the reordered embeddings and two-dimensional Fourier concentration of MLP activations indexed by $(\log_g(a), b)$. For S_4, we decompose representations according to its five irreducible representations and compare the resulting

energy with a random-embedding baseline. These analyses distinguish task-induced algebraic structure from structure attributable to the embedding geometry alone.

The representation analyses are performed on three single-seed checkpoints: the progressive modular-multiplication model, a three-pass causal S_4 model trained with per-pass supervision, and the seeded-sequential S_4 model trained through $k = 8$. The same analysis procedures are applied to each checkpoint.

3.5 Circuit Localization and Causal Patching

We investigate whether modular multiplication is implemented by a localized subset of MLP neurons using a six-layer NanoGPT-scale Looped-GPT trained from scratch. The six-layer Transformer block stack is shared across four recurrent passes. A dedicated token is used for each residue in $\mathbb{Z}_{113}$, so each input and output occupies a single token.

For each physical layer ℓ, let $a_\ell \in \mathbb{R}^d$ denote the MLP activation at the final token. We compute a Fisher score for each neuron with respect to the modular-product output class $c = (a \times b) \bmod 113$, pooling scores across recurrent passes. Within each physical layer, neurons above the 80th percentile define the candidate set S_ℓ.

We then test the causal role of these neurons using activation patching. A base example (a, b) produces product c, while a matched source example (a', b) shares the second operand but has $a' \neq a$, producing product c'. During the source and base forward passes, the activations of neurons in S_ℓ at the final token are copied from the source computation into the corresponding physical layer of the base computation across recurrent passes. We measure both the change in probability assigned to the source answer token and the fraction of examples for which the top-1 prediction switches from c to c'.

4 Experimental Setup

Unless otherwise stated, models are trained with online batches of 512, AdamW, a one-cycle learning-rate schedule with 5% warmup, gradient-norm clipping at 1.0, and curriculum learning in sequence length. A curriculum stage advances after the specified accuracy threshold is reached subject to the minimum optimization-step requirement for that stage.

Modular multiplication. We train three independently initialized models (seeds 0, 1, and 2) with learning rate 10^{-4}, weight decay 0.1, and $(\beta_1, \beta_2) = (0.9, 0.98)$. Training begins at $k = 2$ and advances after 95% final-step training accuracy is reached, with at least 20,000 optimization steps per stage. Training is capped at 500,000 steps and is stopped after ten validation checks without improvement at the final curriculum stage. Checkpoints are selected using a fixed validation set of 2,048 examples at $k = 9$ generated with seed 42. Extrapolation is evaluated on 2,048 fresh examples at each $k = 8, \ldots, 32$, using the nominal number of recurrent steps associated with each access scheme. For modulus 113, random-guess accuracy is $1/112 \approx 0.89\%$. The carry-reset intervention is applied to the seed-0 checkpoint, and recurrent carry norms are measured on 512 fresh examples.

Fixed-Point Reasoning Model. We train three independently initialized FPRM models for 20 epochs on 500,000 pre-generated eight-factor modular products. Models use AdamW with learning rate 10^{-4}, weight decay 0.01, $(\beta_1, \beta_2) = (0.9, 0.95)$, and a constant learning rate. Each final checkpoint is evaluated on 2,048 fresh length-32 examples generated with seed 42 for up to 48 adaptive iterations. Let a_j denote the accuracy of prefix j. We report mean prefix accuracy $A_k = \frac{1}{k}\sum_{j=1}^{k} a_j$. Final-product accuracy is recovered as $kA_k - (k-1)A_{k-1}$, while exact-sequence accuracy requires every prefix product to be correct.

Non-abelian group composition. We train 36 models covering three access schemes, four groups, and three random seeds. Models use the NoPE architecture described above and are trained for 300,000 steps with learning rate 3×10^{-4}, weight decay 0.05, $(\beta_1, \beta_2) = (0.9, 0.98)$, a 90% curriculum threshold, and at least 15,000 steps per curriculum stage. Training proceeds through $K_{\max} = 6$. Checkpoints are selected after mastery at $K_{\max}$ using held-out accuracy at $K_{\max} + 1$.

Extrapolation is evaluated on 4,096 fresh examples at each $k = 6, \ldots, 24$. Reported final-prefix accuracies are three-seed means with two-sided 95% Student-t confidence intervals clipped to $[0, 1]$.

Matched recurrent-update sweep. We compare the standard recurrent update with the Picard update using $\alpha = 0.5$ across five operand-access patterns: one operand, two operands, four operands, all-seen operands, and all operands. All ten conditions use approximately 800,000-parameter models with residual width 256, four attention heads, and MLP width 1024. Operand identity is encoded using sinusoidal codes with scale 0.2. Each k-factor example receives k recurrent updates. Each condition is trained for 300,000 steps with seeds 0–2, learning rate 3×10^{-4}, weight decay 0.05, $(\beta_1, \beta_2) = (0.9, 0.98)$, and 15,000 curriculum steps at each $k = 2, \ldots, 6$, followed by uniform length sampling. Checkpoints are selected using 2,048 held-out examples at $k = 6$ and evaluated on 4,096 fresh examples at each $k = 2, \ldots, 12$. Results are reported as three-seed means with two-sided 95% Student-t confidence intervals clipped to $[0, 1]$.

Representation analysis. Recurrent states are collected from held-out examples at each computation step. Linear probes use Adam with learning rate 0.01 for 100 epochs and are trained and evaluated on disjoint activation partitions. Logit-lens analyses report the most confident neuron and the confidence distribution across neurons. The modular-multiplication Fourier analysis uses primitive root $g = 3$, while the S_4 decomposition excludes the equals token.

Terminal-stability diagnostics. The hold intervention is applied to the progressive multiplication model after its $k = 16$ computation and to the seeded-sequential S_4 model after its $k = 8$ computation. The FPRM and causal three-pass S_4 checkpoints instead receive additional ordinary iterations of their respective trained update rules. These experiments distinguish stability under a fixed computation from continued computation under the model's learned update dynamics.

Circuit analysis. The Looped-GPT is trained from scratch with a six-layer NanoGPT-scale Transformer and four recurrent passes. Fisher scores are computed from last-token MLP activations across recurrent passes, and candidate neurons are selected above the 80th percentile within each physical layer. Causal patching uses matched source and base examples that share the second operand and differ only in the first operand. Patching effects are measured relative to the corresponding unpatched base computation.

5 Results

5.1 Progressive masking supports limited loop-count extrapolation

We first test whether a looped Transformer can learn a reusable iterative update that extends beyond its training horizon, using the progressively masked multiplication model. Across three seeds, progressively masked multiplication reaches 100% final-product accuracy at the training horizon $k = 8$ and at $k = 9$. Mean accuracy falls to 53.7% at $k = 16$, 25.7% at $k = 20$, and 0.85% at $k = 32$, near the $1/112$ chance level (Table 1). At $k = 16$, the individual seeds span 12.5–87.2%. As a diagnostic, resetting the carry in the seed-0 checkpoint reduces $k = 12$ final-product accuracy from 99.9% to 0.9%, showing reliance on information passed across loops (Appendix B).

5.2 Final-product and mean-prefix accuracy diverge in the fixed-point model

The progressively masked model above shows limited but real extrapolation. We next evaluate whether adaptive halting extends this behavior further. Across three final-epoch FPRM checkpoints, mean final-product accuracy falls from 97.3% at $k = 8$ to 14.8% at $k = 20$ and 5.0% at $k = 24$ (Table 1). The corresponding logged mean-prefix accuracies are 74.4% and 63.4%, which average the easier early products with the final one. Exact-sequence accuracy is 3.7% at $k = 20$ and 0.3% at $k = 24$. Per-seed results and the metric derivation appear in Appendix B.

Table 1: Three-seed mean accuracies (%). Progressive and FPRM use separate protocols. Mean prefix averages products 1 through k.

Measure	8	12	16	20	24	32
Progressive final	100.0	93.3	53.7	25.7	10.0	0.85
FPRM final	97.3	82.2	43.9	14.8	5.0	1.2
FPRM mean prefix	99.4	96.2	87.1	74.4	63.4	48.1

5.3 Matched S_4 runs favor one-operand access near the training horizon

The masking-scheme comparison above shows that sequential access outperforms causal access, but it does not isolate the effect of operand count. We therefore vary operand access directly with the operand-access-width sweep (one, two, four, all-seen, all). All 30 matched S_4 runs master the $k = 6$ training horizon. At $k = 7$, one-operand versus full access yields mean accuracy of 0.738 versus 0.080 with the standard looped update, and 0.9998 versus 0.074 with under-relaxed Picard. Every paired seed favors one-operand access. At $k = 8$, the corresponding means are 0.371 versus 0.043 and 0.505 versus 0.041. The three-seed 95% interval for the standard looped one-operand mean at $k = 7$ is wide (clipped to $[0, 1]$), and all access conditions approach the $1/24$ chance level by $k = 10$ (Appendix B.2).

5.4 Readout margin erodes under loop-count extrapolation

Sections 5.1–5.3 establish that accuracy degrades beyond the training horizon. We next ask whether this degradation reflects a destabilized recurrent state or a gradual loss of readout separation. In an independent progressively masked diagnostic checkpoint, the equals-position carry norm stays near 30 through extrapolated depths, while the top-two readout margin falls from about 16.5 early in the computation to 6.7 at depth 18, or roughly 0.6 logits per additional multiplication. Traces at matched depths align across sequence lengths, linking the declining class separation to recurrence depth (Fig. 2). Trained-range S_4 and fixed-point margin measurements appear in Appendix B.3.

5.5 Terminal-state stability is model-dependent

The margin analysis above shows readout separation eroding gradually with depth. We complement it with a direct stability test that examines how a converged model responds to additional computation beyond its usual stopping point. In the independent progressive-mask diagnostic, one extra update with no new factor after a $k = 16$ computation lowers accuracy from approximately 93% to near zero. On separately selected trained-horizon checkpoints, eight additional standard FPRM iterations retain 98.14% accuracy. In contrast, one extra causal pass lowers the three-pass S_4, $k = 3$ model from 100% to 71.21%, while one additional equals-self-only hold lowers the seeded-sequential S_4, $k = 8$ model from 100% to 32.52%. These model-specific interventions reveal substantial differences in output stability (Figure 3, Appendix B.3).

5.6 Operand access determines when masking supports extrapolation

Having isolated operand count from masking family in the matched S_4 sweep above, we now return to the masking-scheme comparison itself (progressive, sequential, and causal) and extend it across four non-abelian groups. This tests whether the progressive-masking failure observed for multiplication (Sec. 5.1) and the sequential scheme's near-horizon advantage generalize beyond a single group.

Across all four non-abelian groups, sequential and causal computation learn the training range, whereas progressive masking remains near chance. At $K_{\max} = 6$, both successful schemes attain at least 0.9999 mean accuracy on every group. The progressive means remain near chance at 0.083 for A_4, 0.090 for D_6, 0.043 for S_4, and 0.044 for D_{12}. Their 95% confidence intervals contain the respective chance levels of $1/12$ or $1/24$, and the curves remain flat as sequence length increases (Figure 4). Thus, progressive masking fails to learn the task under the tested setup, before length extrapolation is evaluated.

Beyond the training horizon, sequential access is consistently more stable than causal access. At $k = 7$, both remain essentially perfect on every group, but by $k = 10$ the sequential means are 0.974

on A_4, 0.908 on D_6, 0.890 on S_4, and 0.942 on D_{12}. The corresponding causal means fall to 0.316, 0.150, 0.286, and 0.351. Uncertainty is small for A_4 and D_6 but wide for S_4 and D_{12} with only three seeds. For example, the $k = 10$ D_{12} interval spans the full $[0, 1]$ range. Despite this uncertainty, the same qualitative pattern appears across all four groups, with sequential access degrading more slowly than causal access.

Overall, the masking comparison shows that progressive masking fails to learn the non-abelian training task under the tested setup, while sequential and causal access both achieve near-perfect accuracy through the training horizon. Among the successful schemes, sequential access retains higher accuracy farther beyond the training horizon.

5.7 Linear probing reveals distinct representations across models

Sections 5.1–5.6 establish that operand access and recurrent state shape whether and how far a model extrapolates. We now examine how models with identical trained-horizon accuracy differ internally. We analyze intermediate representations in three trained models spanning modular multiplication and S_4 composition, distinguished by pass count and masking scheme (Table 2). All reach 100% accuracy at their trained lengths, but differ in how information is exposed and supervised across loops.

Model	Task	Masking	Loops	Accuracy (chance)
Length-gen. mult.	8-factor prefix, $p = 113$	Progressive	7	100.00% (0.89%)
S_4, $k = 3$ three-pass	Non-abelian composition	Causal	3	100.00% (4.17%)
S_4, $k = 8$	Non-abelian composition	Seeded sequential	7	100.00% (4.17%)

Table 2: Models used in the representation experiments.

The multiplication model's probe accuracy forms a clean diagonal (Figure 5). Each loop recovers the prefix it just computed at 100%, while the other prefixes remain at chance (0.89%). Thus, the recurrent state does not retain earlier prefixes in a linearly decodable form.

The three-pass model shows a different pattern. Each prefix reaches 100% accuracy when it is first supervised and remains decodable afterward. By the third pass, the first two prefixes remain at 98.41% and 98.75%, respectively (Figure 6). The final product, however, remains at chance until pass three, even though the causal mask gives the model access to all three factors from the start.

The $k = 8$ model falls between these two cases. Earlier prefixes gradually lose decodability across subsequent loops rather than being immediately erased or fully preserved (Figure 7). The first factor is an exception, remaining 99.69% decodable after seven loops.

5.8 Logit lens: single-neuron confidence varies among successful models

Linear probing shows that the three models encode intermediate products differently. We next examine whether this difference is also reflected in individual neurons. Despite all three models reaching 100% accuracy, logit-lens confidence varies only modestly across models (Table 7 in Appendix C, Fig. 8, 9, and 10). The most confident neuron reaches 3.2× chance in the three-pass model, 3.9× chance in the multiplication model, and 3.3× chance in the $k = 8$ model.

The two S_4 models also differ in how this selectivity is distributed across neurons. In the $k = 8$ model, confidence is spread across a broad, roughly unimodal distribution, while the three-pass model has most neurons only slightly above chance with a thin high-confidence tail (Figures 10 and 9). In both cases, the result is consistent with information being distributed across many weakly selective neurons rather than concentrated in a small set of highly selective units.

5.9 Fourier and representation-theoretic decomposition

The probe and logit-lens analyses above characterize how information is stored and read out across loops. We next examine whether the models also differ in the algebraic structure of their embeddings. For the length-generalizing multiplication model, the embedding spectrum is diffuse rather than

dominated by a single-frequency clock representation. Its strongest embedding frequency explains only 8.5% of the non-DC energy (Figure 11), showing that length generalization is compatible with a diffuse spectrum. The S_4 models show a different structure. Here, energy must be read against the random baseline $d_\rho^2/|G|$, which is 37.5% for each three-dimensional representation (Table 8 in Appendix C). The four named components account for between 73.85% and 77.07% of the total energy, compared with 83.3% for a random embedding. Thus, the structural difference lies in how energy is distributed between the two three-dimensional representations. The three-pass model concentrates 61.93% of its energy in standard×sign while assigning only 1.21% to the plain standard representation (Figure 12). Similarly, the $k = 8$ model concentrates 57.32% in standard×sign and assigns 10.98% to standard (Figure 13), showing a consistent preference for standard×sign across both models.

5.10 Grokking and Circuit Localization in Looped-GPT

Next, we consider whether a looped transformer trained from scratch forms a causally identifiable circuit for modular multiplication, and whether looping changes when that circuit emerges. We train Looped-GPT (6 shared layers, width 384, 10.7M parameters) over 4 loop steps on multiplication mod $p = 113$, against a standard transformer with identical architecture but no looping. We score MLP neurons by how well their activations separate answer classes (Fisher score), then patch the top-scoring neurons in a base input with activations from a source input with a different answer, and measure how often the prediction flips.

Both models grok at nearly the same point and reach 1.000 held-out accuracy, converging at epochs 6,791 (Looped-GPT) and 6,788 (standard transformer), so looping neither accelerates nor delays the transition despite four block passes per forward pass. Patching the top 20% of neurons by maximum Fisher score at every loop step (Table 3) lowers the mean base-answer probability from 0.998 to 0.670, raises the source answer from near zero to 0.213, and flips 22.0% of predictions. The results suggest that answer-relevant representations are distributed across a broader set of neurons rather than localized to the selected subset.

	Looped-GPT
Neurons patched (of 9,216)	461
Base prob. (clean → patched)	0.998 → 0.670
Source prob. (clean → patched)	0.000 → 0.213
Flip rate	22.0% (44/200)

Table 3: Causal intervention results for modular multiplication ($p = 113$).

6 Conclusion

We studied how looped Transformers extend iterative computation beyond the depths encountered during training. Across commutative and non-abelian composition, our results show that extrapolation depends on how operands are exposed and information is carried across recurrent steps, rather than on recurrent depth or training-length accuracy alone.

First, task success and length generalization are separable. Models can master the training horizon yet fail to extrapolate, while models with similar training performance can generalize to different depths. Operand-access patterns therefore shape how far a learned recurrent computation can extend beyond training.

Second, extrapolation failure does not necessarily reflect recurrent-state instability. In a length-generalizing model, the recurrent state remains stable while the readout margin gradually erodes with depth, while terminal-state interventions reveal that continued updates can destabilize otherwise correct solutions. Representation analyses further reveal different patterns of task-relevant information organization among successful extrapolating models, suggesting that behavioral success does not uniquely specify how the solution is represented.

Together, these results show that recurrent depth and training-length accuracy alone do not explain how looped Transformers extrapolate. Operand access, recurrent-state dynamics, and readout also affect extrapolation beyond the training horizon, making these factors important to consider when studying recurrent computation beyond training.

REFERENCES

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Mostafa Haghir Chehreghani. Persistent gaussian perturbations prevent oversmoothing in recurrent graph neural networks. *arXiv preprint arXiv:2607.28185*, 2026.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Pantelis Elinas and Edwin V. Bonilla. Addressing over-smoothing in graph neural networks via deep supervision. *arXiv preprint arXiv:2202.12508*, 2022.

Brady Exoo, Alberto Bietti, and John Sous. Assign and add: A mechanistic study of compositional arithmetic. *arXiv preprint arXiv:2605.31497*, 2026.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, 2025.

Rao Fu, Zixuan Yang, Jiankun Zhang, Jing Ma, Hechang Chen, Yu Li, and Yi Chang. Simply stabilizing the loop via fully looped transformer. *arXiv preprint arXiv:2605.18797*, 2026.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *Proceedings of the 41st International Conference on Machine Learning*, pp. 15130–15152, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 11398–11442, 2023.

Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier circuits in neural networks and transformers: A case study of modular arithmetic with multiple inputs. In *International Conference on Artificial Intelligence and Statistics*, 2025.

Ziming Liu, Ouail Kitouni, Niklas S. Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. *arXiv preprint arXiv:2606.18206*, 2026.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Huu Danh Nguyen. The discrete-log clock: How a transformer learns modular multiplication. *arXiv preprint arXiv:2606.17399*, 2026.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Ricardo Buitrago Ruiz and Albert Gu. Understanding and improving length generalization in recurrent models. *arXiv preprint arXiv:2507.02782*, 2025.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A LIMITATIONS

We collect here the caveats that qualify our results, several of which are noted locally in Section 5.

Seed coverage. Most headline accuracy numbers are means over three seeds, and per-seed variation is often large: progressively masked final-product accuracy at $k = 16$ spans 12.5–87.2% across the three multiplication seeds (Appendix B.1), the three-seed 95% interval for one-operand access at $k = 7$ in the matched S_4 sweep is wide enough to be clipped to $[0, 1]$ (Sec. 5.3), and the non-abelian masking comparison has a $k = 10$ confidence interval for D_{12} that spans the full $[0, 1]$ range (Sec. 5.6). All three representation-analysis models (Sec. 5.7–5.9) are analyzed from a single seed each, so we cannot separate genuine model-to-model differences from seed-level idiosyncrasy. In particular, the choice between the standard and standard$\times$sign representations (Sec. 5.9) may partly reflect which seed was trained rather than passes or masking per se.

Metric interpretation. FPRM's logged mean-prefix accuracy averages easier early-position products together with the harder final product, and is therefore substantially higher than final-product accuracy at the same sequence length. We report both to avoid overstating extrapolation (Sec. 5.2).

B EXTENDED LENGTH AND STATE RESULTS

B.1 SEED-LEVEL MODULAR MULTIPLICATION RESULTS

The progressively masked checkpoints were selected on a fixed $k = 9$ validation batch (Table 4). The FPRM results use final-epoch checkpoints and a fixed length-32 test batch. For FPRM, the logged accuracy A_k is the mean over the first k prefixes. Final-product accuracy at position k is recovered as $kA_k - (k-1)A_{k-1}$. Exact-sequence accuracy requires every prefix through k to be correct.

The carry-reset intervention uses a separately analyzed progressive-mask seed-0 checkpoint. Its unablated pass is perfect at all seven training loops. With the carry reset before each block application, loop zero remains at 100% because it sees the first two factors, while loops one through six fall to 0.6–1.2%, approximately chance. At $k = 12$, the unablated final loop achieves 99.9% and the reset-state pass 0.9%.

Table 4: Final-product accuracy of three progressively masked checkpoints trained through $k = 8$ (2,048 seed-42 test examples per length).

k	Seed 0	Seed 1	Seed 2	Mean
8	1.0000	1.0000	1.0000	1.0000
12	0.9995	0.8359	0.9634	0.9329
16	0.8716	0.1255	0.6123	0.5365
18	0.7085	0.0493	0.4077	0.3885
20	0.5039	0.0239	0.2417	0.2565
24	0.2266	0.0146	0.0586	0.0999
28	0.0586	0.0137	0.0137	0.0286
32	0.0103	0.0054	0.0098	0.0085

Table 5: FPRM accuracy across three final-epoch checkpoints. Final-product accuracy is recovered from the cumulative mean-prefix curve. Mean-prefix and exact-sequence columns average over the three seeds.

k	Final s0	Final s1	Final s2	Mean final	Mean prefix	Exact sequence
8	0.9365	0.9927	0.9893	0.9728	0.9938	0.9678
12	0.6938	0.8726	0.8999	0.8221	0.9619	0.7860
16	0.3916	0.3486	0.5762	0.4388	0.8710	0.3167
18	0.2651	0.1475	0.3599	0.2575	0.8075	0.1229
20	0.1772	0.0566	0.2095	0.1478	0.7444	0.0373
22	0.1279	0.0259	0.1118	0.0885	0.6858	0.0094
24	0.0781	0.0151	0.0562	0.0498	0.6335	0.0028
28	0.0312	0.0068	0.0225	0.0202	0.5472	0.0000
32	0.0156	0.0103	0.0088	0.0116	0.4807	0.0000

B.2 Matched S_4 operand-exposure sweep

All 30 matched runs reach 100% validation and test accuracy at $k = 6$. The four endpoint conditions appear in Table 6. Figure 1 includes the intermediate access widths. Every paired seed favors one-operand over full access at $k = 7$ and $k = 8$. The standard looped one-operand mean at $k = 7$ has a wide clipped 95% interval of $[0, 1]$, so its estimated advantage varies across seeds. At $k = 7$, mean accuracy decreases with access width under both update rules. At $k = 8$, this ordering persists for Picard while the standard looped intermediate widths cross. All access conditions are near chance by $k = 10$.

B.3 Margin and continuation diagnostics

In an independent progressive-mask diagnostic sweep, the equals-position carry norm ranges from 30.0 to 30.8 over the seven training loops and from approximately 28.8 to 30.9 over 23 loops at $k = 24$. The mean top-two logit gap falls from about 16.5 early in computation to 10.8 at depth 14 and 6.7 at depth 18. Once the top prediction is wrong, this top-two gap no longer measures separation from the correct answer.

Additional trained-range checkpoints show distinct margin patterns. In the causal three-pass S_4, $k = 3$ model, mean correct-class margins are 18.07, 17.97, and 18.73 over the three supervised passes (4,147 held-out triples). In seeded-sequential S_4, $k = 8$, the margin rises from 17.62 at the first productive loop to 21.38 at loop 7 (4,096 fresh examples). In a separately selected fixed-point checkpoint, the margin falls from 80.94 for p_1 to 55.85 for p_8 at iteration 32, with 98.14% p_8 accuracy. This last measurement compares prefix positions at fixed compute. Its checkpoint differs from the FPRM runs in Table 5.

For the independent progressive-mask hold test, accuracy is approximately 93% after a standard $k = 16$ computation and falls to near zero after one update with no new factor. In separately selected trained-horizon checkpoints, fixed-point p_8 accuracy remains 98.14% after eight usual iterations beyond reference iteration 32. Causal three-pass S_4, $k = 3$ accuracy falls from 100% to 71.21% after one extra ordinary pass and 7.26% after eight. Seeded-sequential S_4, $k = 8$ accuracy falls from 100% to 32.52% after one equals-self-only hold and 4.00% after eight. After the first extra update, correct-class margins change from 18.73 to 4.86 for $k = 3$ and from 21.38 to -2.12 for $k = 8$.

Table 6: Mean final-product accuracy over three seeds in the matched S_4 grid (4,096 test examples per seed and length). The four displayed conditions have equal parameter counts and use exactly k recurrent updates.

Update	Equals-token access	$k = 6$	$k = 7$	$k = 8$	$k = 10$
Looped	One operand	1.000	0.738	0.371	0.052
Looped	All operands	1.000	0.080	0.043	0.039
Picard	One operand	1.000	0.9998	0.505	0.045
Picard	All operands	1.000	0.074	0.041	0.043

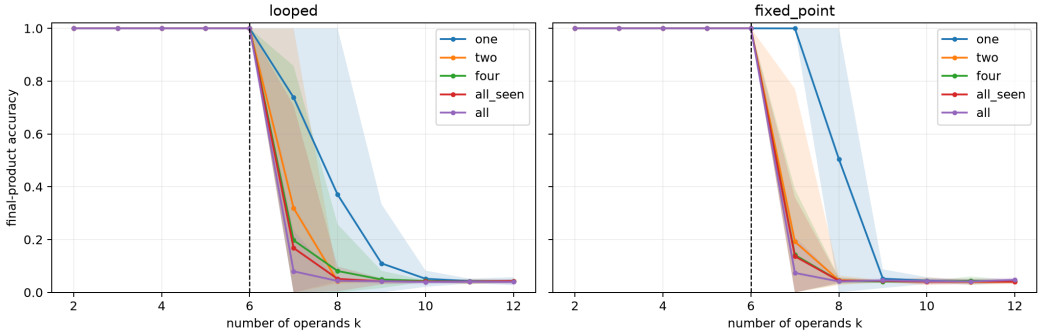


Figure 1: Matched S_4 operand-exposure sweep: standard looped update (left) and parameter-matched Picard update with relaxation 0.5 (right). Curves show three-seed means and clipped 95% Student-t intervals. The dashed line marks the $k = 6$ training horizon. All-seen access exposes the observed prefix, while full access exposes every factor at each loop.

C Additional group and representation figures

Model	Top-neuron confidence	× chance
Length-gen. mult.	3.45%	3.9×
S_4, $k = 3$ three-pass	13.54%	3.2×
S_4, $k = 8$	13.74%	3.3×

Table 7: Top-neuron logit-lens confidence across the three trained models.

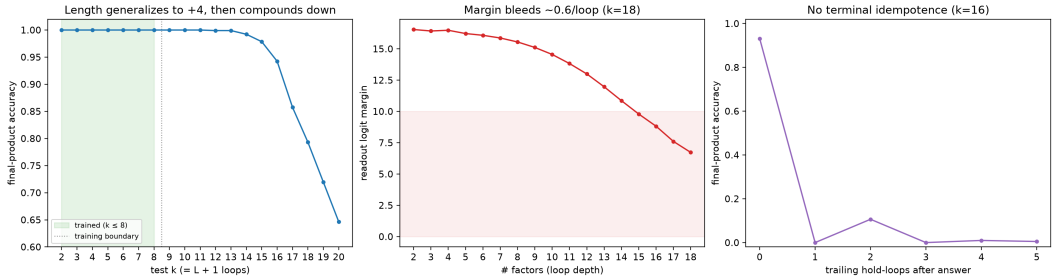


Figure 2: Independent progressive-mask diagnostic sweep: accuracy versus length (left), readout margin versus loop depth (center), and accuracy after a terminal hold update following a 16-factor product (right).

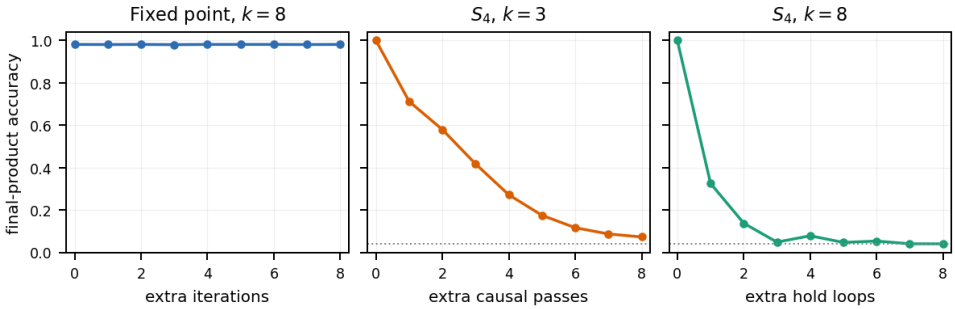


Figure 3: Final-product accuracy under extra updates for separately selected seed-0 checkpoints. Zero is the trained answer readout: iteration 32 for fixed point, pass 3 for causal S_4, $k = 3$, and loop 7 for seeded-sequential S_4, $k = 8$. Continuation interventions differ by model. Dotted lines show S_4 chance accuracy $(1/24)$.

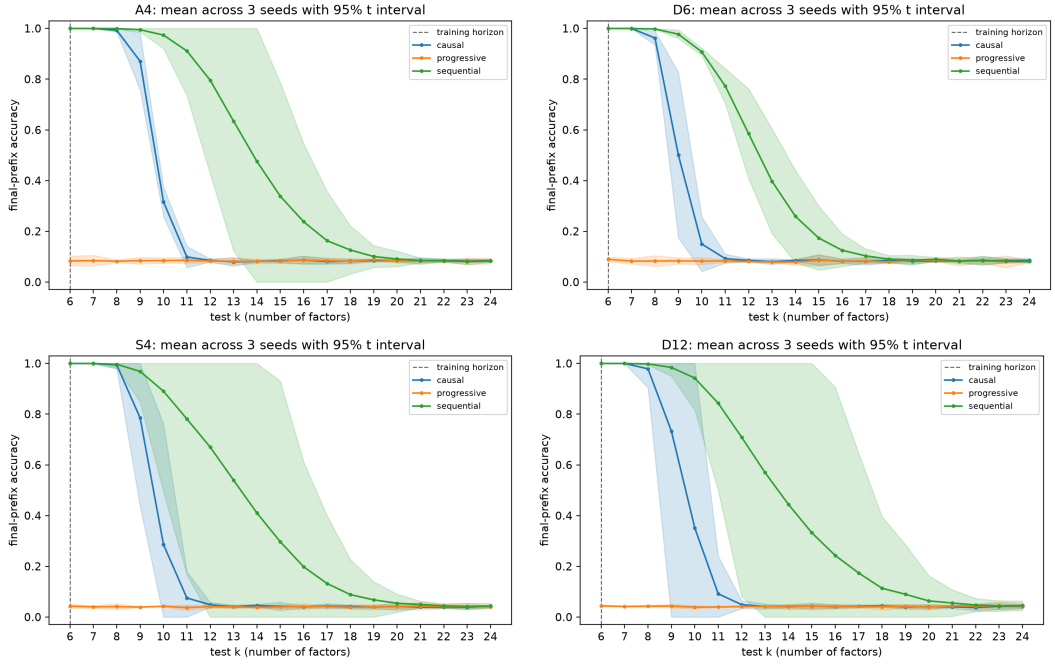


Figure 4: Final-prefix accuracy for non-abelian group products. Curves show means over three seeds with shaded two-sided 95% Student-t intervals. The dashed line marks the training horizon $K_{\max} = 6$.

Representation	Dim.	Baseline	$k = 3$ three	$k = 8$
Trivial	1	4.17%	3.70%	2.80%
Standard	3	37.50%	1.21%	10.98%
Sign	1	4.17%	7.01%	5.97%
Standard $\times$ Sign	3	37.50%	61.93%	57.32%
Two-dimensional	2	16.67%	26.15%	22.93%
Sum of first four		83.33%	73.85%	77.07%

Table 8: Representation-theoretic decomposition of the token embeddings of the two S_4 models. The baseline is the expected energy share of an embedding without group structure, $d_\rho^2/|G|$. The two-dimensional energy is the remainder. Each model concentrates on one three-dimensional representation and suppresses the other, and the sum of the first four components stays below the baseline.

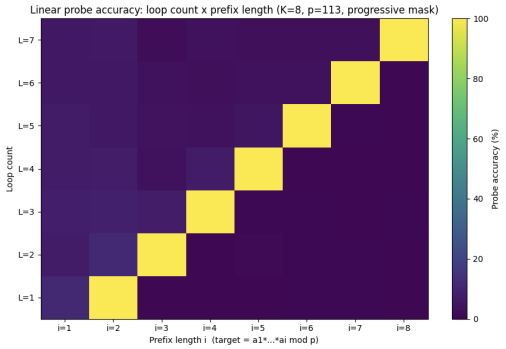


Figure 5: Linear probe accuracy across loop count and prefix length for the length-generalizing multiplication model. Each loop recovers its corresponding prefix product at 100%, while other prefix lengths remain near chance (0.89%).

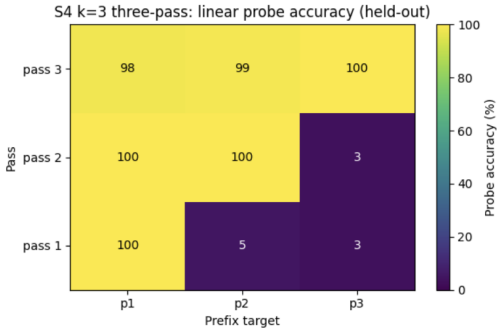


Figure 6: Linear probe accuracy across passes and prefix targets for the three-pass S_4 model ($k = 3$), on held-out inputs. Each prefix is recovered at the pass where it is supervised and retained afterward, while the final product stays at chance until the third pass.

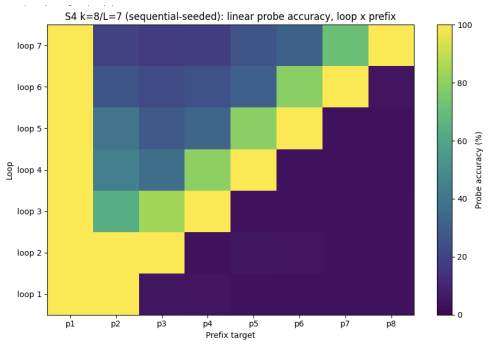


Figure 7: Linear probe accuracy across loop count and prefix target for the $k = 8$ non-abelian composition model under seeded-sequential masking. Earlier prefixes decay gradually across loops, while the seeded first element (p1) remains near 100%.

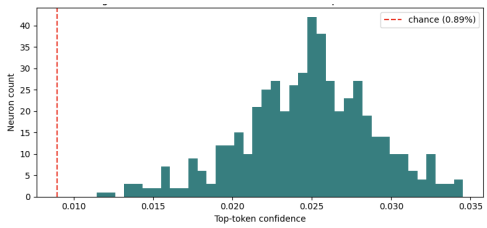


Figure 8: Top-neuron logit-lens confidence for the length-generalizing multiplication model. Chance accuracy is 0.89%.

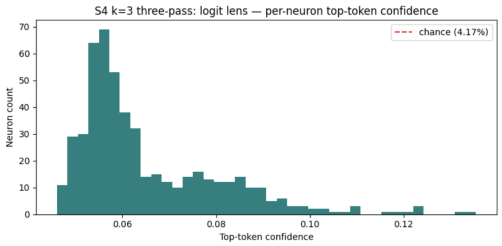


Figure 9: Distribution of top-token confidence across neurons for the three-pass S_4 model ($k = 3$). Chance accuracy is 4.17%.

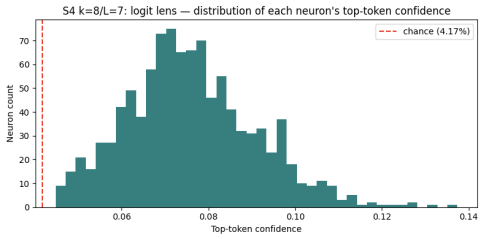


Figure 10: Distribution of top-token confidence across neurons for the $k = 8$ non-abelian composition model. Chance accuracy is 4.17%.

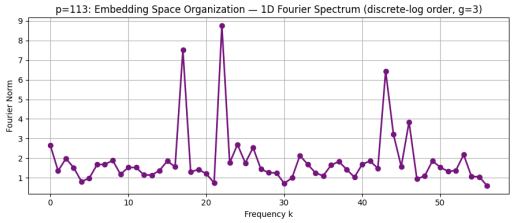


Figure 11: 1D Fourier spectrum of the embedding for the length-generalizing multiplication model.

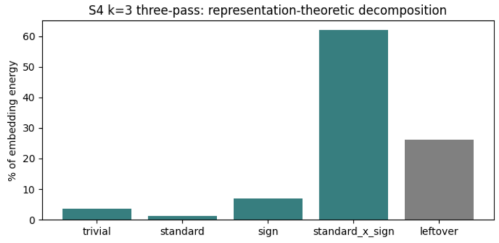


Figure 12: Representation-theoretic decomposition of the token embeddings for the three-pass S_4 model ($k = 3$). The gray bar is the two-dimensional representation.

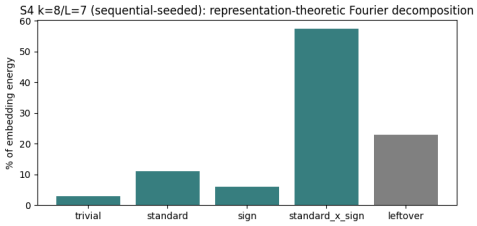


Figure 13: Representation-theoretic decomposition of the embedding for the $k = 8$ non-abelian composition model.