How Training Loop Counts Shape Computation in Looped Transformers

Anonymous authors
Paper under double-blind review

Abstract

A looped transformer applies the same block repeatedly before producing an answer. We ask how the number of loops used during training changes the computation learned by that block. We study sequence tasks with a known correct output at each position. Reading these outputs after each loop reveals a computation frontier that advances from the start of the sequence. Its average advance per loop is frontier speed. For cumulative products of permutations of four objects, models trained with fewer loops show faster measurable frontiers. This relation holds when models share initial weights and training examples and receive the same number of optimizer updates. It also appears when accuracy determines advancement through a length curriculum. We then continue training two copies of each source model. Reducing the loop count increases frontier speed relative to keeping it fixed. Replacing intermediate states disrupts later outputs. Additional loops can extend correct computation to the last position and then reduce the fraction of fully correct answers. These findings show how training shapes progress inside a shared block and how that progress determines when its output is useful.

1 Introduction

A looped transformer applies the same neural block repeatedly to an evolving state. More loops add computation without adding parameters (Dehghani et al., 2019; Geiping et al., 2025; Zhu et al., 2025). The training loop count specifies how many applications precede the loss. In our main experiments, a fixed loop multiplier m sets $T(n) = \max\{2, \lceil mn \rceil\}$ for input length n. This schedule is chosen before training and determines the count for every input of that length. Fewer loops give the model fewer state updates in which to produce the same output sequence. Finishing sooner requires faster progress or a shorter initial delay.

Yang et al. (2024) show that the training loop count affects convergence in in-context regression. Studies of length generalization (Fan et al., 2025), compositional reasoning (Kohli et al., 2026), and pretrained looped language models (Blayney et al., 2026) examine how answers and states change with recurrence. These results raise a training question. With the same block and task, does prescribing fewer loops teach each update to accomplish more work? Can progress per loop change after learning? Each evaluation loop adds computation, so choosing when to read a correct answer matters for inference cost. We track partial progress, complete answers, and their persistence.

Consider running parity. Each output reports whether the input from the first position through the current position contains an odd number of ones. For input $(1, 0, 1)$, the outputs are $(1, 1, 0)$. Reading the first bit gives 1. The next bit leaves the value at 1. The final bit changes it to 0. A sequential algorithm updates one parity bit using exclusive OR. A learned model can distribute this work across positions and loops. We ask whether its repeated updates carry intermediate work through to a correct answer.

After every loop, we read the model's outputs on a fixed evaluation batch. Starting at the first position, we find how far the model meets an accuracy threshold at every position. This boundary is the computation frontier. It may advance at different speeds across loops or temporarily retreat. Frontier speed v is its average advance over a measured interval. Frontier completion is the first loop at which it reaches the last position. Exact-match accuracy counts an input as correct only when every output is correct. Figure 1 shows a running parity example. Figure 2 reports the measured trajectories.

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.

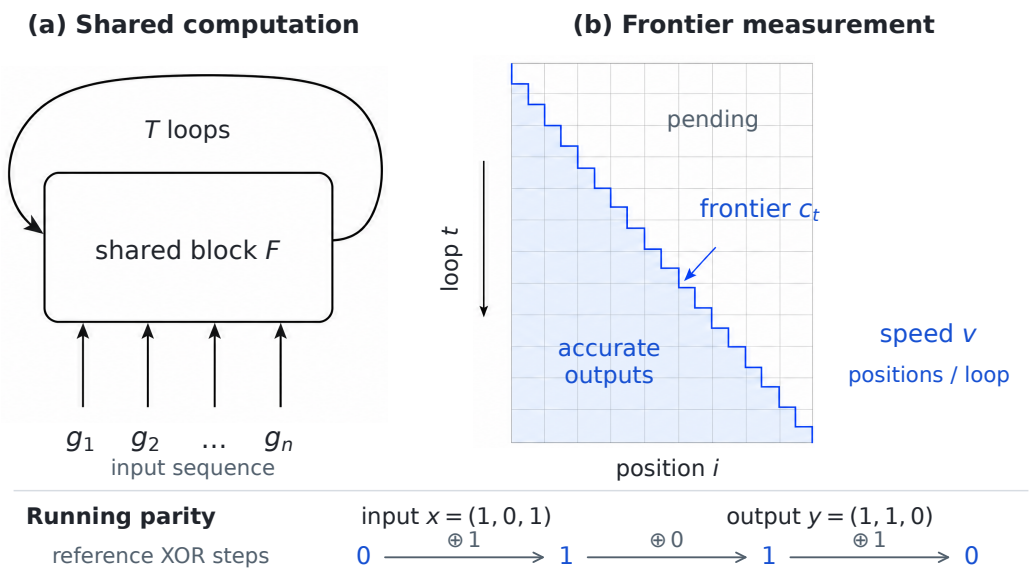

Figure 1. The shared block updates the sequence state at each loop. Running parity maps input $(1, 0, 1)$ to output $(1, 1, 0)$. Every position from the start through the frontier meets the accuracy threshold. Its average advance per loop defines frontier speed.

Running parity updates a bit through exclusive OR. Decimal increment carries a digit change along a sequence. Cumulative group products repeatedly compose operations. These tasks are simple enough to specify every correct output exactly. They let us observe partial computation as well as the final answer. Our main comparison uses S_4, the permutations of four objects. We change the training loop count while keeping the task and architecture fixed. Matching the initial weights and training examples separates this change from differences in initialization or data.

Our contributions are three findings.

1. **Models trained with fewer loops show faster measurable frontiers.** The relation holds with matched optimizer-update counts and also appears under an adaptive curriculum. The final-loop loss shapes intermediate progress without supervising each loop separately.
2. **Continued training can change frontier speed.** From identical source weights, reducing the loop count increases speed relative to keeping the original schedule. Increasing the count reduces speed where both paired models have measurable frontiers.
3. **Intermediate states support later answers, but more loops can lower accuracy.** State replacements disrupt downstream outputs. Further loops can reach full coverage and then reduce exact-match accuracy. Correct intermediate outputs and a correct final answer must therefore be measured separately.

2 How we observe computation

Setting the training loop count. Let n be the number of task tokens, excluding the beginning-of-sequence token. The training loop count $T(n)$ is the number of times the shared block processes an input before the training loss is computed. The main S_4 experiments set $T(n) = \max\{2, \lceil mn \rceil\}$, where the fixed loop multiplier m sets loops per position. For a given m, the count depends only on input length. The inverse $d = 1/m$ is the nominal demand. It expresses the required positions of progress per loop before rounding; the realized demand is $n/T(n)$. Higher demand means fewer loops for the same input. We write T when the length is fixed. The loss uses all supervised positions after $T(n)$ loops. One optimizer update then changes the parameters using gradients from back-propagation. At evaluation, we read outputs at $T(n)$ and after further loops. Table 3 summarizes the training comparisons.

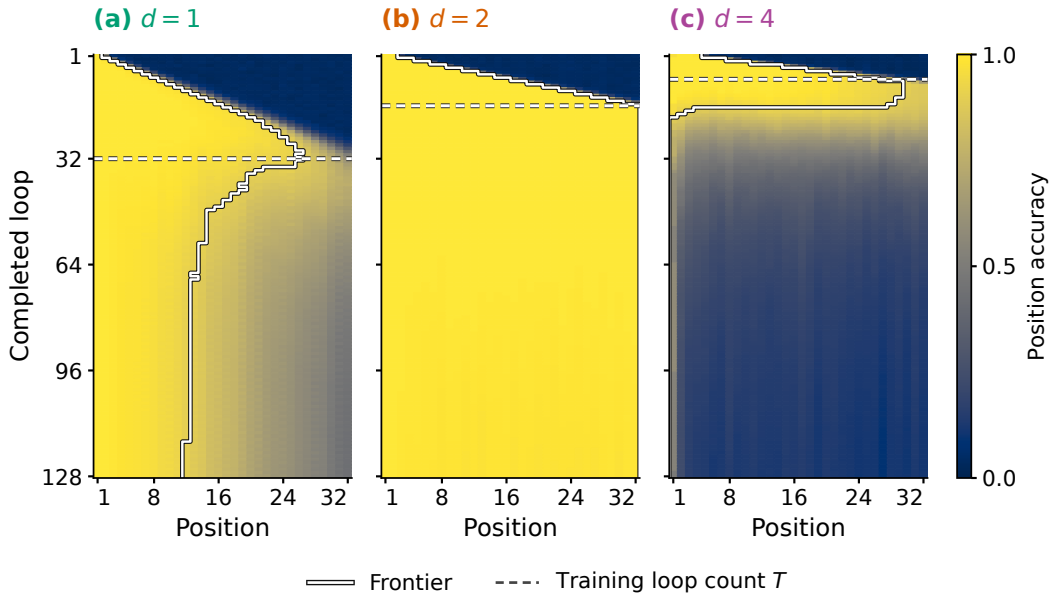

Figure 2. The frontier advances differently under different training loop counts. These S_4 models share initialization, training examples, and evaluation inputs. Axes and accuracy scale match across the complete $4n$ window. The solid line marks the frontier. The dashed line marks T. Loops increase downward. Appendix A.3 gives the selection rule.

Defining the correct output. The running parity algorithm in Section 1 maintains a bit s_i with $s_0 = 0$ and $s_i = s_{i-1} \oplus x_i$ for input bits x_i. Its output is $y_i = s_i$. In group theory, a product combines elements using the group's operation. For operation $\circ$, input $g_1, \ldots, g_n$ has correct output $y_i = g_1 \circ \cdots \circ g_i$ at each position. In $\mathbb{Z}_{60}$ the operation is addition modulo 60. In permutation groups it is composition. The groups S_4 and S_5 contain all permutations of four and five objects; A_5 contains the even permutations of five objects. These permutation tasks are order-sensitive even though composition is associative. Running parity and decimal increment supply additional task comparisons. Their generators, supervised positions and algebraic assumptions appear in Appendix B.

Let e be the input embedding and h^t the state after t completed loops. The model applies

$$h^{t+1} = F_\theta(h^t, e), \qquad \hat{y}_i^t = \arg\max_c [W \,\mathrm{LN}(h_i^t)]_c. \tag{1}$$

W is the shared readout and LN is layer normalization. The main S_4 block has two pre-norm layers, width 128 and four attention heads. It uses causal attention without positional embeddings. A learned adapter supplies the input embedding at every loop. Appendix B gives the full architecture.

Tracking progress across loops. On a fixed evaluation batch, let $a_i(t)$ be the fraction of correct outputs at position i. We define

$$c_t(n) = \max\{k \in \{0, \ldots, n\} \mid a_i(t) \geq .9 \text{ for all } i \leq k\}. \tag{2}$$

The frontier position $c_t(n)$ counts consecutive positions meeting the threshold from the first task token. Full coverage means $c_t(n) = n$. Frontier coverage is $c_t(n)/n$. Exact-match accuracy is the fraction of inputs for which every output position is correct. One wrong position makes that sequence incorrect under this measure. Per-token accuracy still counts each correct position. Full coverage requires every position to meet the accuracy threshold across the evaluation batch. Full coverage can still include incorrectly predicted sequences. All measurements exclude the beginning-of-sequence token.

Let b be the first loop attaining the largest recorded frontier position, and a the loop before the first recorded increase in c_t. If at least two positive increments occur by b, the frontier speed is

$$v_{\text{first}} = \frac{c_b - c_a}{b - a}. \tag{3}$$

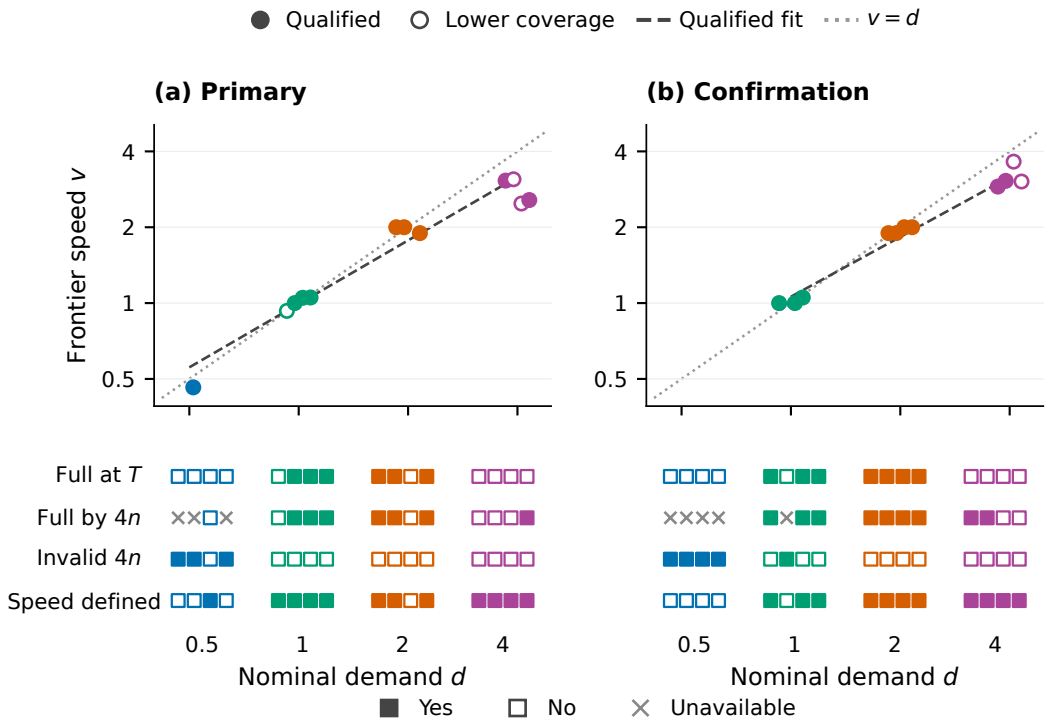

Figure 3. Fewer training loops accompany faster progress at fixed optimizer-update count. Each model receives 30,000 optimizer updates. Colors identify demand d. Filled circles meet the speed-fit criterion in Section 2; Open circles have lower coverage. Each strip shows four seeds from one cohort at each demand. Input length is $n = 32$. Filled/open squares mean yes/no; crosses mean unavailable. The rows are independent outcomes. Speed is median v_{first} over both evaluation lengths. Evaluations at T include models with invalid longer trajectories.

This first-peak speed averages the advance from just before the first increase to the first attainment of the observed maximum. The last-increment speed v_{last} ends at the last positive increment in the observation window. It includes later stalls, retreats, and renewed growth. The two definitions summarize different parts of the same trajectory. Both training comparisons observe $H = 4n$ loops. A model needs a defined speed at both lengths 16 and 32; its reported speed is their ordinary median. The fit includes models with measurable speeds and peak frontier coverage of at least .85 at both lengths. We call these models qualified. The completion measurements include every assigned model.

The same shared architecture supports different patterns of progress. Figure 2 holds initialization, task, and training examples fixed while changing the loop count. The shared axes expose when the frontier starts moving and how its position changes. Speed summarizes its initial frontier progress; the full trajectory retains completion and later answer changes. The comparisons below test this pattern across training seeds. Appendix F describes the measurements available after a state has become fixed.

3 Training with fewer loops

We now vary the number of training loops while keeping the input task fixed. The horizontal axis is nominal demand $d = 1/m$. Our first comparison gives every model the same number of parameter updates. We then test an adaptive curriculum, where accuracy determines when training advances to longer inputs.

Table 1. The exponent relating speed to demand depends on the speed definition. Entries give β in $v = Ad^{\beta}$ for the same qualified models in each row. The two cohorts contain separate training seeds.

Training	Cohort	Qualified	First peak	Last increment
Fixed optimizer updates	Primary	9/16	0.837	0.884
	Confirmation	9/16	0.789	0.789
Adaptive curriculum	Primary	13/16	0.929	1.462
	Confirmation	13/16	0.871	1.325

Matching the number of optimizer updates. We compare demands $d \in \{.5, 1, 2, 4\}$ using eight paired training seeds. Four seeds form the primary cohort and four form the confirmation cohort. All 32 models complete 30,000 optimizer updates in the same four stages. Initial weights and training examples match within a seed. Evaluation uses 1,024 common inputs per length from a separate random stream. Computational work per input varies with the loop count.

Fewer training loops accompany faster measurable progress. The fitted speed increases with demand in both cohorts (Figure 3). Table 1 compares the fitted exponents under the two speed definitions. The figure also shows models outside that fit. In the confirmation cohort, the setting with the most training loops has no qualified frontier. Its outcomes remain visible alongside the other settings.

The rows below the speed plots show when computation reaches full coverage. They distinguish full coverage at T, first attainment within the longer window, and numerical failure. Some models develop nonfinite states during the longer evaluation. Evaluating those models only to T gives finite outputs but no correct complete answers. Appendix A.3 gives the procedure. These outcomes show how far and when the frontier advances.

Advancing to longer inputs as accuracy improves. The second experiment lets each model advance to the next curriculum stage when it reaches the accuracy criterion. The loop multiplier m stays fixed within each condition. It compares $d \in \{.5, 1, 2, 4\}$ across eight seeds, split equally into primary and confirmation cohorts. Architecture, optimizer, data generation, and advancement rules are shared. The stage lengths and update limits are given in Appendix A.3. Because a stage can end early, models may receive different numbers of parameter updates. This comparison tests the speed relation under a common training rule that responds to learning progress.

The increasing speed relation also appears with the adaptive curriculum. Table 1 reports the fits for both cohorts. Figure 4 tracks frontier completion across all assigned models. Dividing the evaluation loop by the training loop count places each model's training cutoff at the same point on the horizontal axis.

The fitted exponent depends on how progress is measured. Table 1 compares the speed definitions on the same qualified models. The largest changes occur in models trained with more loops under the adaptive curriculum. Their frontier coverage can alternate between one and zero after first completion. Each return to full coverage creates another positive increment. The first-peak speed ends at the first observed maximum; the last-increment speed includes these later changes (Figure 7). Both fitted relations increase with demand, but their exponents differ. Repeating the measurement with fresh inputs processed in smaller batches preserves that direction. Appendix A.3 reports the models included in each fit and the resulting estimates.

Reaching the last position. Speed is only one component of completion time. For progress $c_t \simeq v(t - t_0)$ starting at loop t_0, completion by T requires $v(T - t_0) \gtrsim n$. A model must advance fast enough to finish in the available loops. Selecting only successful models would favor speeds above this threshold. We therefore compare the fitted relation with completion outcomes for every assigned model. Steps beyond the training-loop cutoff in Figure 4 identify models whose first full coverage appears after T loops. A model can have a frontier speed close to demand yet finish after the cutoff. The time spent before propagation begins can contribute to this delay. Speed describes how fast the frontier advances. Completion timing records when it reaches the last position.

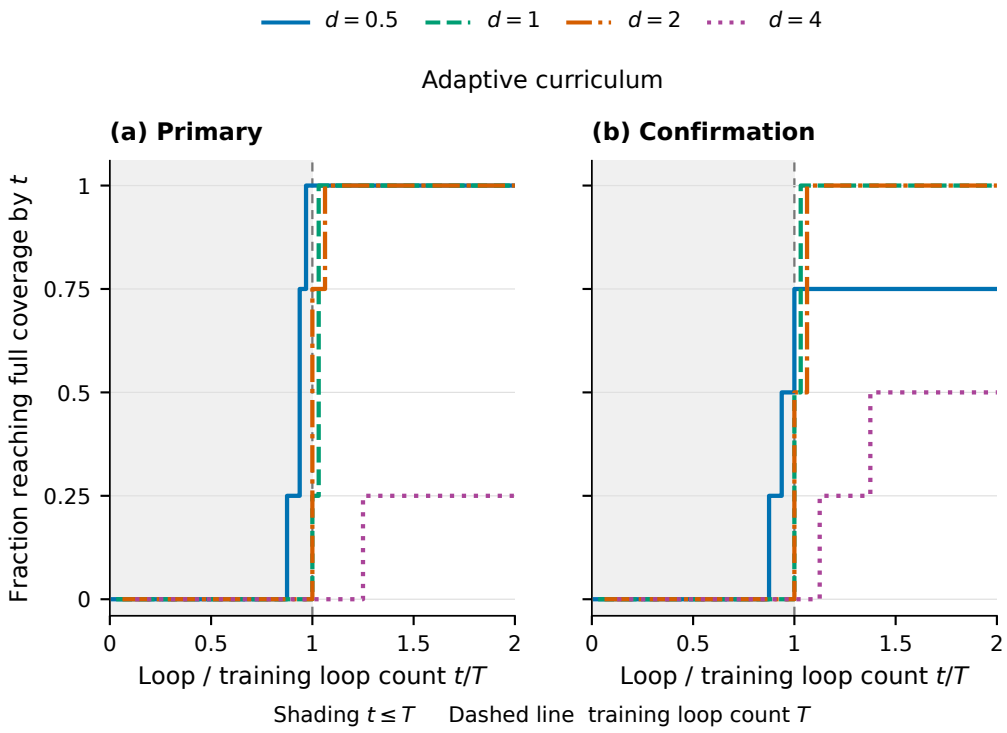

Figure 4. Full coverage can first occur after T loops. Each curve gives the fraction of all four adaptive-stage S_4 models per group and demand that have reached full coverage by loop t, at $n = 32$. The shared interval extends to $2T$; shading ends at T. The curves accumulate first attainment. Figure 7 tracks subsequent accuracy.

The running parity example makes this timing concrete. A sequential procedure with k compositions per loop and s_0 startup loops finishes at $s_0 + \lceil n/k \rceil$. A readout restricted to particular loops can delay the answer further. An associative scan instead combines segments in parallel.

The broader variable-curriculum comparison in Appendix A.2 gives an approximately proportional relation using the last-increment speed. Its common-window analysis also gives an increasing relation under both speed definitions. Across these comparisons, frontier speed increases with demand. The fitted exponent depends on the training and measurement settings.

4 How intermediate computation changes

Replacing intermediate states. An advancing frontier shows progress. We next test whether the intermediate state contributes to later answers. After loop t, we replace the state at position i with a state from another input at that same position and loop. The model then executes the remaining loops. Figure 5 uses one A_5 checkpoint and one S_4 checkpoint. At length 32, we evaluate 32 loops for A_5 and 18 loops for S_4. Two evaluation batches per checkpoint provide recipient examples whose complete outputs are correct before replacement. Both batches evaluate the same trained checkpoint. Appendix A.4 reports the recipient and patch counts.

Replacing intermediate states disrupts later answers in a structured pattern. In the S_4 model, the strongest effects move toward later positions as replacement occurs at later loops. The A_5 effects are weaker and more diffuse under the same color scale. The checks that leave the state unchanged preserve predictions. Outputs before the replaced position also stay unchanged, as expected from causal attention. Changes occur in later outputs. Their distribution across positions and loops shows where the intermediate state contributes to the answer.

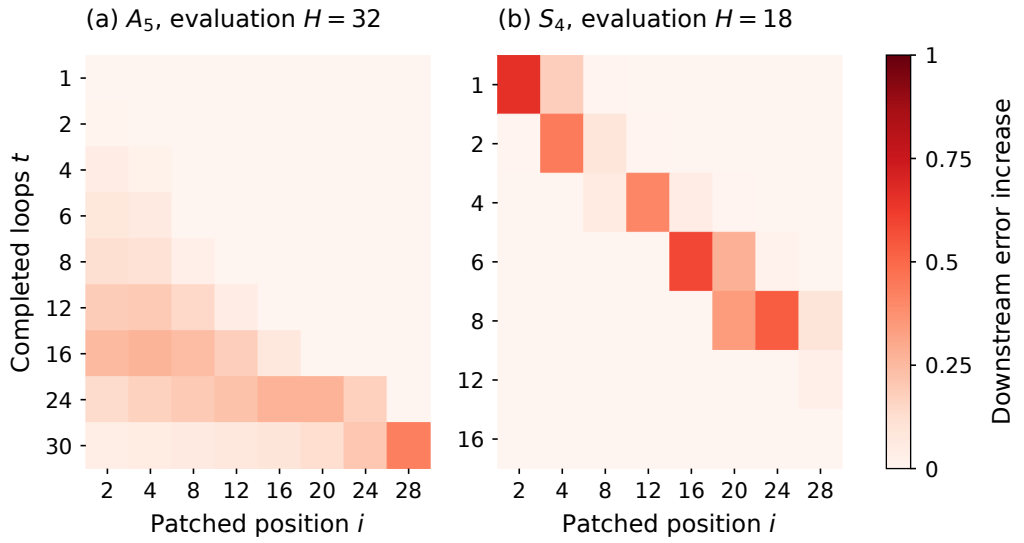

Figure 5. Replacing an intermediate state disrupts later answers. Panels show one A_5 and one S_4 model. Color gives the increase in error rate at positions after the replacement, among inputs whose complete output was initially correct. Each cell pools error increases and eligible token counts across two batches before division. Both panels share a color scale. Donors are different inputs. Unmodified forward passes and no-change checks preserve predictions. Details appear in Appendix A.4.

Changing the training loop count. For each trained source model, we create two copies and continue training both. One keeps the loop count, called stay; the other changes it, called switch. We test both directions between demands one and two, using four source seeds per direction. Each copy receives 7,000 optimizer updates. The first stage uses lengths up to 16 for 3,000 optimizer updates. The next uses lengths up to 32 for 4,000 optimizer updates. The copies start from identical weights, see identical examples, and use fresh AdamW optimizer states. They differ in the loop count used for each input.

Changing the training loop count changes frontier speed. Reducing the loop count increases speed in every pair. Increasing it reduces speed in pairs where both copies have a measurable frontier. Figure 6 shows each pair and its success at T. The models trained with fewer loops succeed in the evaluated cohort; those trained with more loops have mixed success. The figure also shows conditions with undefined speeds.

Comparing against stay separates the effect of changing the loop count from the effect of simply continuing training. The two directions show that speed can change while exact-match accuracy follows a different pattern.

5 When to Read the Answer

Evaluating with additional loops. Figure 4 records when the frontier first reaches the last position. We now track exact-match accuracy as looping continues, using exact-match and per-token accuracy beyond the training loop count. A cumulative completion curve can only rise. Once a model has reached full coverage, it remains counted even if its later outputs change. An accuracy curve can fall because it evaluates the output at each loop. Reading the curves together shows when progress first covers the input and how useful the model's output remains afterward. **More loops can first complete an answer and then damage it.** Figure 7 averages every assigned model within each adaptive-curriculum cohort and demand. At demand two, exact-match accuracy falls when evaluation continues beyond the training cutoff. Per-token accuracy remains much higher. The model continues to produce many correct local answers while losing complete solutions. Measuring both quantities reveals how additional iterations change the usefulness of the computation.

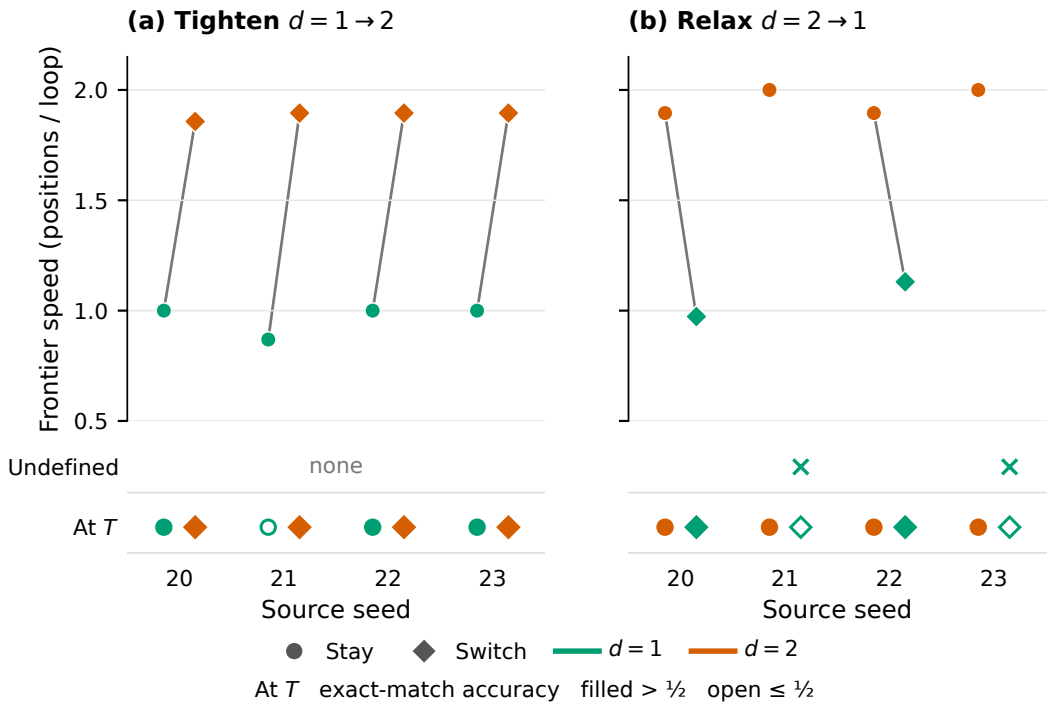

Figure 6. Changing the training loop count changes frontier speed. Each line connects stay and switch training from the same source model, with four sources per direction. Colors identify target demand. The copies share training examples and optimizer-update counts; recurrent work differs. Crosses mark undefined speeds. At each copy's own T, filled/open markers mean exact-match accuracy $> .5/\leq .5$ on $1{,}024$ inputs. Table 7 gives summaries.

The timing of the readout also matters. For $d = .5$, outputs at even loops can be accurate while exact-match accuracy at adjacent odd loops is zero in both cohorts. The training readout occurs at $T = 2n$, which is always even. Figure 7 shows both branches. Evaluating only the even loops would give a different account of answer stability from evaluating every loop. A stopping rule therefore needs to specify when an output is read.

Testing on longer inputs. We compare frontier speed and frontier coverage at T with accuracy on longer inputs. Frontier coverage is $c_T(n)/n$. Three further scores describe the trajectory at $n = 16$. Adjacent-state cosine compares consecutive hidden states. Jacobian-vector-product (JVP) alignment compares how the shared block transforms the same small perturbation at successive states. Output agreement compares predictions from different initial states.

Each score ranks the models. Spearman correlation compares that ranking with the ranking by longer-input accuracy. AUROC measures how well it separates models above the cohort median accuracy from the rest. Table 2 reports these comparisons at lengths 64 and 128 under the training loop schedule, using the available pairs of scores and outcomes. **The frontier association is positive for the group tasks and near zero for running parity.** Higher JVP alignment accompanies lower longer-input accuracy on S_4. Its raw-score AUROC reflects this reversed ranking. Unavailable A_5 outcomes remain missing. The comparisons use score directions assessed on these same models.

6 What Training Changes

The running parity algorithm stores the answer so far and updates it with each new bit. Our measurements examine corresponding properties in learned models. The frontier tracks partial progress, exact match tests the complete answer, and further loops test whether it stays correct. A stopping rule specifies when to read the output, including its phase when answers alternate across loops.

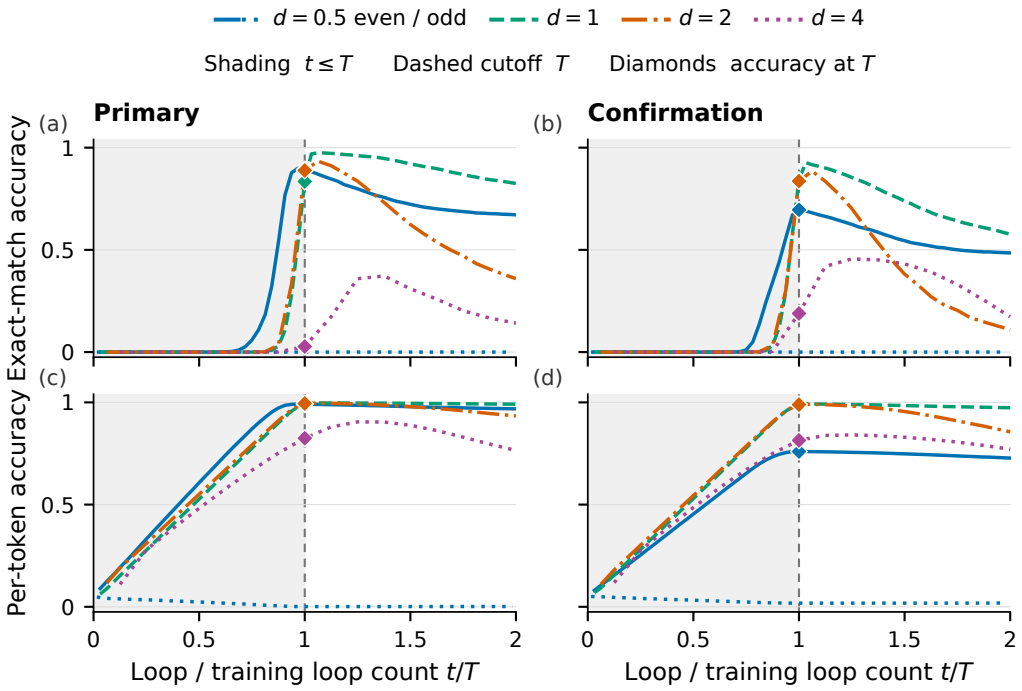

Figure 7. Additional loops can reduce exact-match accuracy. Each curve averages four models within a cohort and demand. Evaluation uses 1,024 paired inputs at $n = 32$. Colors identify demand. Shading ends at T. Diamonds mark accuracy at T. For $d = .5$, even- and odd-loop outputs are separate. Training uses an even loop count.

Table 2. Associations with longer-input per-token accuracy. Entries give Spearman ρ, with tie-corrected raw-score AUROC in parentheses. Complete pairs number $16/7/16$ for S_4/A_5/running parity; test lengths are $64, 128$. Frontiers use $n = 32/32/16$, respectively. Other metrics use $n = 16$. Appendix A.6 gives the cohort definitions and training lengths. Constant running-parity coverage gives undefined ρ and AUROC .50. Output agreement is measured across initializations.

Metric	S_4 log-T	A_5 $T{=}n$	running parity $T{=}n$
frontier speed	$+0.81$ (0.82)	$+0.63$ (1.00)	-0.08 (0.61)
frontier coverage at T	$+0.80$ (0.76)	$+0.95$ (1.00)	n/a (0.50)
cosine (adjacent)	-0.00 (0.45)	$+0.50$ (0.67)	-0.04 (0.36)
sampled-JVP alignment	-0.65 (0.09)	$+0.64$ (0.75)	-0.04 (0.44)
output agreement	$+0.51$ (0.66)	$+0.45$ (0.67)	$+0.13$ (0.53)

Training loop count changes how fast the frontier advances. Matched training and paired continuations link loop count to frontier speed. State replacements connect intermediate work to later answers. Completion and continued accuracy show when this work yields a useful output. These tests ask whether a model carries partial work through to a correct answer under a specified stopping rule across the evaluated lengths.

7 Conclusion

Training loop count shapes frontier speed in the tested looped transformers. Matched comparisons and paired continuations show that fewer loops yield faster measurable frontiers. State replacements show that intermediate states affect later answers. Additional loops can reduce exact-match accuracy after correct answers have appeared. For models with alternating outputs, the readout phase also determines answer quality.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and coding assistance. The authors reviewed the resulting text and code and take responsibility for the final content of this work, including its text, code and claims.

ETHICS STATEMENT

Our experiments use synthetic sequence tasks and a task from the public easy-to-hard benchmark. Arithmetic and group-composition programs define the correct outputs. The study involves no human participants or personal data. Third-party datasets and software retain their original distribution terms.

REPRODUCIBILITY STATEMENT

Section 2 defines the models and tasks. It also defines the frontier measurements and training loop schedules. Sections 3, 4 and 5 describe the experiments. Appendix B gives the training settings. Appendix A describes the evaluations. Appendix F gives assumptions and proofs. The supplementary material contains model configurations and evaluation code. It also contains numerical summaries and verification scripts. These identify the evaluated cohorts and observation windows.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, et al. Circuit tracing: Revealing computational graphs in language models. transformer-circuits.pub, 2025.

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger Grosse. Path independent equilibrium models can better exploit test-time computation. *NeurIPS*, 2022a.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *NeurIPS*, 2022b.

Artur Back de Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. *ICML*, 2024.

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. *NeurIPS*, 2025. arXiv:2507.10524.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *NeurIPS*, 2019.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. *NeurIPS*, 2022.

David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, 1988.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. *EMNLP*, 2020.

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.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. *Findings of ACL*, 2024.

Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. Iteration head: A mechanistic study of chain-of-thought. *NeurIPS*, 2024.

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. *ACL*, 2022.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *ICML*, 2023.

Mark M. Churchland, John P. Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012. doi: 10.1038/nature11129.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *NeurIPS*, 2023.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. *ICLR*, 2022.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. *NeurIPS*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *ICLR*, 2019.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the Chomsky hierarchy. *ICLR*, 2023.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, et al. Faith and fate: Limits of transformers on compositionality. *NeurIPS*, 2023.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *ICLR*, 2020.

Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. transformer-circuits.pub, 2021.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. *NeurIPS*, 2023.

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? *ICML*, 2024.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *NeurIPS*, 2021.

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, pp. 41340–41391, 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. *ICML*, 2023.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *ICLR*, 2024.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *TACL*, 2020.

Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *ACL*, 2024. arXiv:2402.09963.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *NeurIPS*, 2023.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. *NeurIPS*, 2022.

Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, et al. A generalist neural algorithmic learner. *Learning on Graphs Conference (LoG)*, 2022.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. *ICLR*, 2016.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *NeurIPS*, 2023.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=8fz7WRThKL.

Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116: 450–464, 1965.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *ICLR*, 2020.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *ICLR*, 2024.

Nuo Li, Kayvon Daie, Karel Svoboda, and Shaul Druckmann. Robust neuronal dynamics in premotor cortex during motor planning. *Nature*, 532(7600):459–464, 2016. doi: 10.1038/nature17643.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *ICLR*, 2024.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. *NeurIPS*, 2023.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *ICLR*, 2023.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *First Workshop on the Application of LLM Explainability to Reasoning and Planning at COLM*, 2025. arXiv:2507.02199.

Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013. doi: 10.1038/nature12742.

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. *NeurIPS*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *NeurIPS*, 2022.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *TACL*, 2023.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *ICLR*, 2024.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *NeurIPS*, 2025. arXiv:2503.03961.

William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *TACL*, 2022.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *ICML*, 2024. arXiv:2404.08819.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *ICLR*, 2023.

nostalgebraist. Interpreting GPT: The logit lens. LessWrong, 2020.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. transformer-circuits.pub, 2022.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. *COLM*, 2024.

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.

Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. *NeurIPS*, 2023.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *ICML*, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *ICLR*, 2025.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *NeurIPS*, 2022.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *NeurIPS*, 2021.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. *ICLR*, 2025.

Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets. *ICML*, 2024.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *TACL*, 2024.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. *EMNLP*, 2023.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The clrs algorithmic reasoning benchmark. *ICML*, 2022.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *NeurIPS*, 2020.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. *NeurIPS*, 2024.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *ICLR*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. *ICML*, 2021.

Liu Yang, Kangwook Lee, Robert D. Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HHbRxoDTxE.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. *NeurIPS*, 2024.

Qifan Yu, Zhenyu He, Sijie Li, Zhou Xun, Jun Zhang, Jingjing Xu, and Di He. Enhancing autoregressive chain-of-thought through loop-aligned reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2206–2222. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-long.97. URL https://aclanthology.org/2026.eacl-long.97/.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *NeurIPS*, 2023.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *ICLR*, 2024.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

Table 3. The comparisons answer different questions. Counts refer to assigned model conditions, not independent evaluation batches. All frontier comparisons use lengths 16 and 32. Reused source weights remain paired.

Comparison	Models	Shared and varied settings	Frontier measurement
Variable curricula	27	Same model scale; curricula and stage update limits vary	Recorded windows, last increment
Adaptive curriculum	32	Eight seeds across four loop schedules; common adaptive promotion rule	Common $4n$, first peak
Fixed updates	32	Eight paired seeds; same initial weights, data and 30,000 updates	Common $4n$, first peak
Stay or switch	16	Four seeds per source loop schedule; same source weights, data and 7,000 updates	Common $4n$, paired first-peak speeds

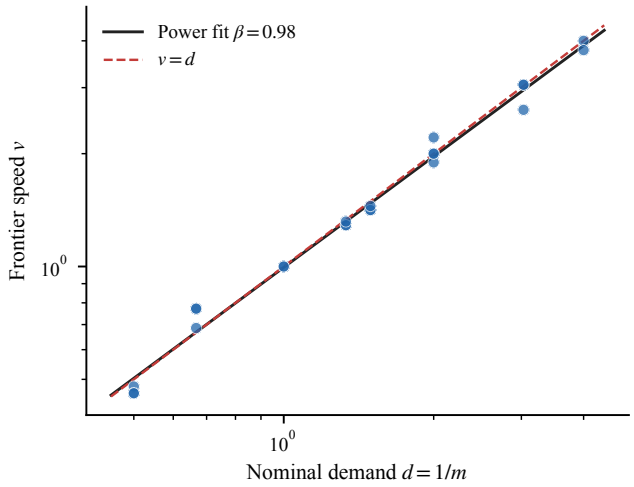

Figure 8. Frontier speed increases with nominal demand across different training settings. The fit uses 23 measurable S_4 models among 27 trained models and nominal demand $d = 1/m$, rounded to three decimals. Curricula differ across models. Figure 4 tracks frontier completion across all assigned models in the adaptive-curriculum comparison.

A Evaluation details

A.1 When outputs become correct and stop changing

Exact-match accuracy is the fraction of inputs with every supervised position correct. Per-token accuracy averages correctness over positions. Both exclude the beginning-of-sequence token. For a fixed observation window H, let $\hat{y}_i^t$ be the label decoded after t completed loops. The finite-window stabilization time is

$$\tau(n, i) = \min\{k \in \{1, \ldots, H\} \mid \hat{y}_i^t = \hat{y}_i^H \text{ for every } t \in \{k, \ldots, H\}\}.$$

This time records when the decoded label settles on its final observed value. Label correctness and hidden-state stability are measured separately. For analyses of correct complete outputs, we select inputs the model solves.

A.2 How frontier speed changes with demand

At loop t, a position meets the criterion when its batch accuracy reaches .9. The frontier position is the last position for which this criterion holds at every position from the start. The last-increment speed measures average advance from the loop before the first positive increment to the last positive increment. It requires at least two such increments.

The regression in Figure 8 uses 23 measurable scale-s S_4 models among 27 attempted multiplier configurations. The families are `s4v2`, `s4v4`, and `s4v7`. Each model contributes the median of

its available frontier speeds at $n \in \{16, 32\}$. The observation windows contain 40 and 64 loops. One $n = 32$ trajectory is observed for 80 loops. Curricula and stage update limits vary across these families. The fit excludes models trained with logarithmic loop schedules or learned positions. It also excludes transferred models and larger scales.

Nominal demand is $d = 1/m$ for the training loop count $T(n) = \max\{2, \lceil mn \rceil\}$. The realized demand is $n/\max\{2, \lceil mn \rceil\}$. It can differ because T is an integer. For this fit, we round $1/m$ to three decimals before regression and keep frontier speeds at full precision throughout aggregation and fitting. The log-space fit is $v = 0.995\, d^{0.983}$. The fitted exponent is 0.98 and $R^2 = 0.99$. The fit includes models with a measurable frontier. None of the three $T = 3n$ models has a measurable frontier speed; three of the four $T = 2n$ models do.

Table 4. Points in the frontier speed fit. Median frontier speed over $n \in \{16, 32\}$ within a model, then across models at each demand. The same 23 model-level frontier speeds determine the regression.

nominal demand $1/m$	0.50	0.67	1.00	1.33	1.49	2.00	3.03	4.00
seeds	3	3	2	3	3	4	3	2
median v	0.46	0.77	1.00	1.29	1.41	2.00	3.05	3.89

A separate common-$4n$ evaluation gives a first-peak exponent of .907 with $R^2 = .983$. Using the last-increment endpoint in that common window gives 1.182 with $R^2 = .932$. These fits show how the speed definition changes the exponent within a common window. The fit across different training settings uses the frontier speeds and observation windows specified above.

Table 5 compares frontier speeds across tasks and training conditions. It includes available frontier speeds at $n \in \{16, 32, 64\}$, extending the length range used in the frontier speed fit. General frontier evaluations use

$$H = \min\{64, \max[16, \lfloor 2.5 T_{\text{base}}(n) \rfloor]\}.$$

For proportional training loop schedules, $T_{\text{base}} = n$ independently of multiplier m. The 2.5 factor scales this base schedule. The depth study uses the threshold on frontier coverage given in Appendix C.3.

Table 5. Frontier speeds by training loop schedule. Each model's frontier speed is the median over available $n \in \{16, 32, 64\}$; the table then takes the median across models [range]. Model counts give those with measurable frontier speeds over all trained models; A_5 additionally requires exact-match accuracy $> .5$ at $n = 32, t = 32$. The last column is the median final frontier coverage at $n = 64$ over the stated cohort, including models without a measurable frontier speed, at $t = 64$ (proportional) or $t = 22$ (log). Frontier speeds summarize the selected interval, including nonlinear motion on $\mathbb{Z}_{60}$. Cohort membership is in Appendix B.3.

Group	Loop schedule (demand n/T)	models	frontier speed	coverage@64
S_4 S	$T = 2n$ (0.5)	3/4	0.51 [0.51, 0.52]	0.54
S_4 S	$T = n$ (1)	2/2	1.00 [1.00, 1.00]	0.72
S_4 S	$T = n/2$ (2)	4/4	2.00 [1.93, 2.13]	0.10
S_4 S	$T = n/4$ (4)	2/2	3.80 [3.60, 4.00]	0.00
S_4 S	logarithmic (mixed)	16/16	2.00 [1.86, 2.42]	0.30
S_4 M	logarithmic (mixed)	4/4	1.88 [1.86, 1.88]	0.36
S_4 L	logarithmic (mixed)	4/4	2.66 [2.09, 3.33]	0.07
A_5 S	$T = n$, horizon	5/16	1.00 [0.86, 1.03]	n/a
S_5 M	op.-first, $T = n$	2/2	1.00 [1.00, 1.00]	0.28
$\mathbb{Z}_{60}$ S	logarithmic (mixed)	8/8	2.08 [1.82, 5.17]	n/a

The models in Figure 9 are evaluated at $n = 32$. The panels use an S_4 checkpoint with a logarithmic training loop schedule, an A_5 checkpoint with a proportional training loop schedule, and an S_4 checkpoint trained with $T = n/2$. Their frontier speeds at this length are approximately 2.36, 1.03, and 2.13. The displayed trajectories contain 20, 64 and 24 loops. Their observation windows contain 20, 64, and 64 loops in the same order. Accuracy matrices are stored to two decimals; frontier positions and frontier speeds keep their original precision. Both axes begin at one and count completed loops and supervised positions.

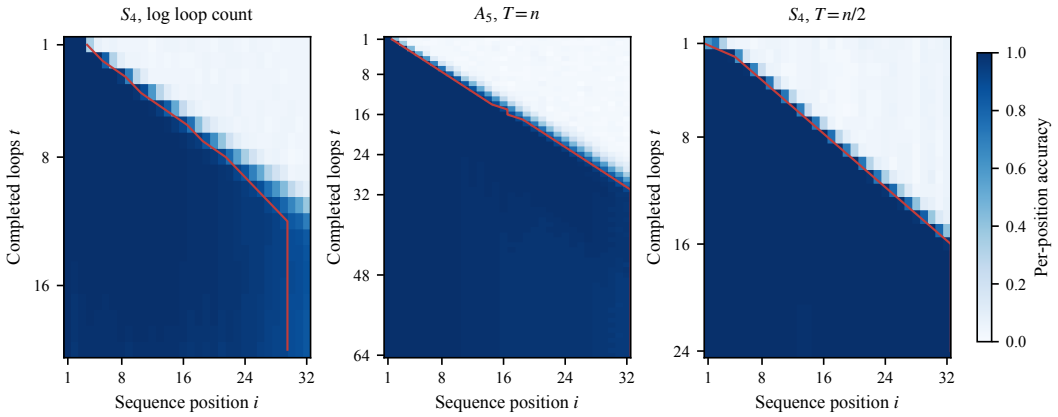

Figure 9. Frontier speeds differ across tasks and training schedules at $n = 32$. The panels show S_4 with a logarithmic training loop schedule, A_5 with a proportional training loop schedule, and S_4 trained with a loop count equal to half the input length. The tasks and observation windows differ. The boundary uses the saved frontier before accuracy-color rounding. Appendix A.2 identifies the trajectories.

A.3 How the training comparisons are evaluated

The adaptive-curriculum S_4 experiment assigns multipliers $2, 1, .5, .25$ to each of seeds 20 through 27. Seeds 20 through 23 form the primary cohort and 24 through 27 the confirmation cohort. All use scale S with input injection and no positional embeddings. Their AdamW settings are the same. The four curriculum stages have maximum lengths $4, 8, 16, 32$ and update caps $6{,}000, 6{,}000, 8{,}000, 10{,}000$. Training advances to the next stage when per-token accuracy reaches .98 and the minimum stage duration has elapsed. The number of optimizer updates therefore varies with loop count and learning progress. Frontier coverage and accuracy analyses include all 32 assigned models, whose saved weights are finite.

Common-window evaluation uses $1{,}024$ inputs per model at each of lengths 16 and 32. The same inputs are used across models and loop counts. Each model is observed for $4n$ loops, and the frontier uses a per-position threshold of .9. To enter the fit, a model must have at least two positive increments by the first peak and peak frontier coverage of at least .85 at both lengths. Models with measurable speeds below this coverage threshold appear in the plot but stay outside the fit. Both speed definitions are fitted to the same models, and undefined frontier speeds remain missing. Outcomes over the full window and at T retain the separate denominators of their evaluated model sets.

The fixed-update experiment uses seeds 40 through 47 with the same four multipliers. Seeds 40 through 43 form the primary cohort and 44 through 47 the confirmation cohort. All stages execute their full caps, totaling $30{,}000$ optimizer updates. Weight initialization, sampled sequence lengths and task instances are paired across the four conditions for a seed. Evaluation uses a separate random stream. Two seeds per cohort use RTX 4090 hardware with PyTorch 2.5.1 and two use RTX 3090 hardware with PyTorch 2.4.0. All four conditions for a seed use the same host, optimizer-update count, and training examples. The loop count changes how often tokens and attention pairs are processed. Both hosts use identical model, generator, and trainer code. We evaluate each model on $1{,}024$ CPU-generated paired inputs at each of lengths 16 and 32, processed in chunks of 128 for $4n$ loops. We check that weights, recurrent states, and readout values are finite. On the first chunk, unmodified forward passes and no-change checks must reproduce the labels exactly. Integer correct counts and per-input correctness records support independent recomputation of both first-correct times and correctness at later loops. Failed training and numerical evaluations remain in the attempted-condition counts, with accuracy marked unavailable.

Which trajectories are shown. Figure 2 compares fixed-update S_4 models at $n = 32$. We display the first primary-cohort seed with valid complete observation windows at demands one, two, and four. Selection depends on window validity, independently of eligibility for the speed fit. The

search follows ascending seed order across the four primary seeds; seed 40 already meets the rule. All three conditions appear, including demand one, which fails the coverage threshold. The panels share 1,024 evaluation inputs, a loop window, and an accuracy scale. Contours apply the frontier threshold to integer correct counts at each position. Each panel marks its own training loop count.

Evaluating models with nonfinite long trajectories. Every fixed-update model finishes 30,000 training updates with finite checkpoint tensors. Twenty-four common-$4n$ evaluations are numerically valid. The other eight fail a recurrent-state finiteness check. Seven are at demand .5 and one at demand one. Their full-window frontier speeds and frontier completion histories remain unavailable. After observing these failures, we defined an evaluation that stops at T. We applied it to every affected model using the same 1,024 inputs. Chunk size, weights, and validation checks also remain unchanged. All eight endpoint evaluations are finite, and each gives $0/1{,}024$ correct complete outputs at both lengths. They provide accuracy at loop T; first-attainment times and longer histories remain unavailable. Thus all 32 models have measured accuracy at T, while full-window frontier speeds and curves use the 24 models with valid trajectories.

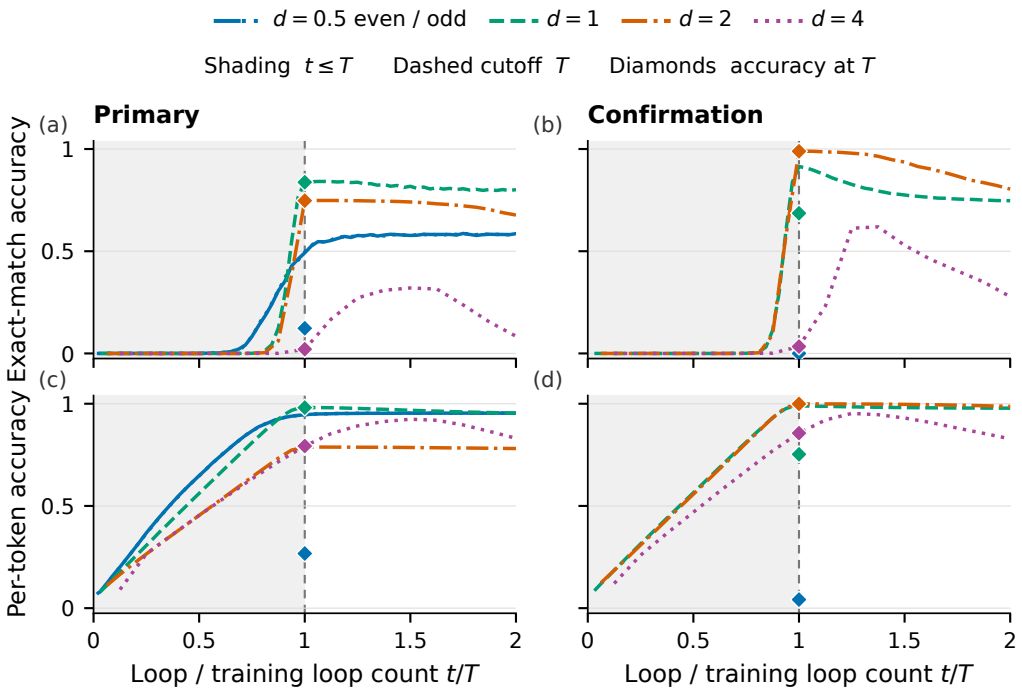

Figure 10. Accuracy beyond the training loop count after fixed-update training. Curves average models with a valid full observation window. Diamonds show accuracy at T, including separate evaluations of models with unavailable longer trajectories. Figure 3 shows each model's trajectory availability. Even- and odd-loop outputs are plotted separately when they differ.

Changing the evaluation inputs and batch size. We reevaluate the same thirty-two models trained with the adaptive curriculum in chunks of 128 with numerical checks and a separate shared input seed. The weights are fixed. Both sets of measurements are retained. Every chunked trajectory is numerically valid. Table 6 compares full-batch and chunked evaluation. One primary-cohort demand-four model changes qualification under the frontier coverage threshold. Its peak frontier coverage at $n = 32$ changes from .8125 to .875 between these evaluations. Both give an exact-match count of $1/1{,}024$ at T. The weights are unchanged. Both the input sample and inference chunking change between these measurements.

Table 6. Full-batch and chunked evaluation of the same weights from adaptive-curriculum training. The full-batch measurement uses a batch of $1{,}024$ inputs. The chunked measurement uses separate paired inputs in chunks of 128. Both use a $4n$ window and the same thresholds. Each fit uses the models that meet the qualification criteria in that evaluation.

Cohort	Evaluation	Qualified	First peak	Last increment
Primary	Full batch	13/16	0.929	1.462
Primary	Chunked	14/16	0.913	1.375
Confirmation	Full batch	13/16	0.871	1.325
Confirmation	Chunked	13/16	0.879	1.314

Table 7. Same-source continuations with a fixed or changed loop count. Median frontier speeds use identical finite stay/switch pairs. All four assigned models per condition have valid evaluations at T. Success requires exact-match accuracy $> .5$ at length 32 under that condition's training loop count. Both continuations use $7{,}000$ updates and paired data; executed recurrent work differs.

Demand change	Stay $\rightarrow$ switch speed	Speed pairs	Stay success	Switch success
$1 \rightarrow 2$	$1.000 \rightarrow 1.895$	4/4	3/4	4/4
$2 \rightarrow 1$	$1.895 \rightarrow 1.052$	2/4	4/4	2/4

A.4 What changes when a state is replaced

Figure 5 evaluates one A_5 checkpoint and one S_4 checkpoint. Their training seeds are 0 and 1, respectively. Both use width 128, four heads, and two layers in the repeated block. The A_5 checkpoint uses a proportional training loop schedule and a horizon curriculum. The S_4 checkpoint uses a logarithmic training loop schedule. Both measurements use $n = 32$. The observation window is $H = 32$ for A_5 and $H = 18$ for S_4. The latter exceeds its training-schedule count at that length. Each panel follows the corresponding model through its specified observation window.

Each checkpoint has two evaluation batches of 256 examples. Both batches evaluate the same trained weights. The A_5 batches contain 241 and 244 examples with a correct complete clean output. The S_4 batches contain 185 and 179 such examples. We replace states in these correctly solved inputs. Donors are drawn from the full batch and paired with recipients at the same position and completed loop. Every pair contains different inputs. Self-pairs are excluded.

The patched positions are $2, 4, 8, 12, 16, 20, 24, 28$. For A_5, patched loop counts are $1, 2, 4, 6, 8, 12, 16, 24, 30$. For S_4, they are $1, 2, 4, 6, 8, 12, 16$. Each patch is followed by the remaining loops. At a patch location (t, i), let $E_b(t, i)$ be the increase in the number of incorrect supervised tokens at positions $j > i$ in evaluation batch b. Let R_b count its clean-correct recipients. The displayed value is

$$D(t, i) = \frac{\sum_b E_b(t, i)}{\sum_b R_b(n - i)}.$$

The recipient outputs are initially correct, so their clean error count is zero. We sum integer error and token counts before division. We exclude the patched position and use no damage threshold. Both panels share a color scale. Repeated patch locations reuse the same recipients.

Standard forward passes and the loop-by-loop evaluator give identical predictions and logits. All no-change interventions leave predictions and final states unchanged. There are 72 no-change cells per A_5 batch and 56 per S_4 batch. Evaluation leaves checkpoint hashes unchanged. Upstream predictions also remain unchanged, as required by the causal attention mask. The comparison therefore focuses on where and when downstream predictions are affected.

A.5 Changing the loop count after training

Table 7 contains sixteen continuations from eight source checkpoints. Seeds 20 through 23 provide four sources at demand one and four at demand two. Each source starts a stay condition and a switch condition with identical initial weights, verified by their hashes. Each completes $7{,}000$ optimizer updates. The two stages have maximum lengths 16 and 32. They receive $3{,}000$ and $4{,}000$ optimizer updates respectively. Both copies use the same data seeds and sampled lengths. They see the

same examples. AdamW starts fresh in both conditions. Example and supervised-token counts match within each pair. Changing the loop count changes how often tokens and attention pairs are processed. Both continuations inherit the same source pretraining.

All sixteen evaluations over $4n$ loops are numerically valid. They use the same numerical checks, chunk size, and inputs as the fixed-update experiment. Success at T means exact-match accuracy greater than .5 on 1,024 inputs at $n = 32$ under the condition's own loop count. Both conditions have endpoint measurements for every pair. A model's frontier speed is the median of its positive first-peak speeds at $n = 16$ and $n = 32$. Paired frontier speed comparisons require both conditions to have a defined frontier speed, yielding four pairs for demand $1 \to 2$ and two for $2 \to 1$. The other two switch conditions remain in the accuracy and assigned-model counts.

The four frontier speed changes after reducing the loop count are positive. The two defined changes after increasing it are negative. Exact two-sided paired sign-flip probabilities are .125 and .5, respectively. Each direction compares four independent source-seed pairs. The eight source checkpoints share these four seed values across the two directions.

A transfer-versus-scratch experiment compares eight transferred models with eight models trained from scratch under the same settings for 7,000 updates. Its paired success at T counts are $4/4$ versus $3/4$ for demand $1 \to 2$ and $2/4$ versus $4/4$ for $2 \to 1$. This experiment compares transfer with training from scratch; Table 7 compares continuations of the same source weights.

A separate schedule-transfer experiment changes a proportional training loop schedule to a logarithmic one. It uses four seeds each for A_5 and S_4. At $n = 32, T = 8$, token accuracies range from .606 to .614 for A_5 and .72 to .73 for S_4. Neither task cohort contains a model with target exact-match accuracy above .5. Reverse S_4 transfer to $T = 32$ succeeds in four of four evaluated models. Frontier trajectories are available for two. These models form a separate cohort from the matched stay/switch comparison.

Choosing an evaluation loop count. The comparison in Figure 11 uses $t^\star = \lceil n/\hat{v} \rceil + s$ with slack $s \in \{0, 2, 4\}$ and plots $s = 2$. The reference evaluation loop count is $t = n$ for every model, including one trained with $m = .5$. The oracle selects the best accuracy over the tested loop counts. The speed-based rule uses a frontier speed measured before evaluation.

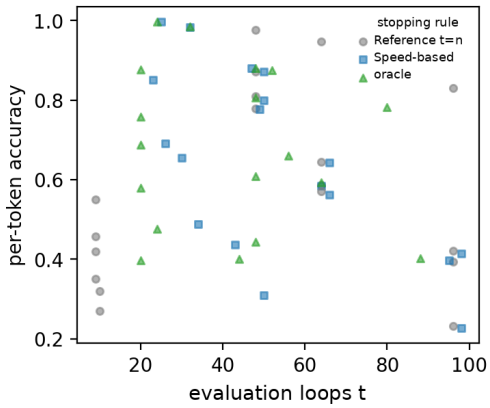

Figure 11. Loop allocation from measured frontier speed. The speed-based rule uses $\lceil n/\hat{v} \rceil + 2$ loops. The reference uses $t = n$, twice the training count when $m = .5$. The oracle selects the highest accuracy within the evaluated loop sweep.

A.6 Does frontier progress track longer-input accuracy?

The counts above/at-or-below median longer-input accuracy are $8/8$, $3/4$ and $8/8$ for S_4/A_5/running parity, respectively. Training lengths are $32/32/16$; test lengths are $64, 128$.

Table 2 relates frontier and similarity measurements to accuracy on longer inputs. Each task uses 16 training seeds with the same settings. Models with both measurements available give 16 pairs for S_4,

7 for A_5, and 16 for running parity. Missing A_5 outcomes remain unavailable. The outcome is mean per-token accuracy at $n \in \{64, 128\}$ under the training loop schedules. Frontier measurements use the training lengths $n = 32$ for group tasks and $n = 16$ for running parity.

Readout measurements use 1,024 examples per model, similarity probes use 128, and each longer-length accuracy measurement uses 1,024. The first-peak speed is measured over $4n$ loops. Its interval ends at the first observed maximum frontier position. It requires at least two positive increments. The fit across different training settings uses the last-increment speed. Frontier coverage $c_T(n)/n$ is measured at the base training-schedule counts $T = 8, 32, 16$ for the three cohorts.

Similarity measurements use $n = 16$. They compare cosine similarity between adjacent states, JVP alignment under a shared perturbation at sampled transitions, and output agreement across different initializations. Output agreement measures behavioral consistency motivated by the path independence of Anil et al. (2022a). We report Spearman correlation with ties handled and AUROC using scores in their original direction. Each task is split at its own median accuracy for the binary AUROC outcome.

How score direction changes AUROC. Reversing a score changes AUROC A to $1 - A$. The sampled-JVP values for S_4 are .09375 and .90625 in the two orientations. The corresponding A_5 values are .75 and .25. The two orientations use the same observations. For prediction, the score direction is fixed before evaluating held-out outcomes.

B How models are trained

B.1 Task outputs and model design

For decimal digits $d_1, \ldots, d_n$ listed least significant first, increment has correct output

$$y_i = \left(d_i + \prod_{j<i} \mathbf{1}\{d_j = 9\} \right) \bmod 10.$$

The empty product equals one. To prevent overflow, the generator places a non-nine digit after the leading chain of nines. Running parity has correct output $y_i = (\sum_{j \le i} b_j) \bmod 2$. Group products use the exact cumulative composition defined in §2.

A group is solvable when repeatedly taking its commutator subgroup reaches the identity subgroup. A commutator $aba^{-1}b^{-1}$ measures failure to commute. S_4 is noncommutative but solvable. S_5 and A_5 are non-solvable. Solvable groups admit constant-depth solutions in the circuit model of Liu et al. (2023). The non-solvable group word problems considered here are NC^1-complete (Barrington, 1989). The cited circuit models specify the precision and computation assumptions for these complexity results.

The model uses pre-norm transformer blocks with hidden width D, feed-forward width $4D$, and GELU activations. Weights are shared across loops. Input injection applies a learned linear adapter to $[h; e]$. The shared readout acts on $\mathrm{LN}(h^T)$. Scale S has two layers with width 128 and four attention heads. It has approximately .44M parameters. Scale M has two layers with width 256 and eight attention heads. It has approximately 1.72M parameters. Scale L has four layers, width 512, and eight heads. It has approximately 13.16M parameters on S_4 and 13.26M on S_5. Parameter counts depend on the vocabulary. The U12 model uses twelve distinct layers repeated for $T = 2$ loops. Its effective depth matches scale S at $T = 12$. The depth comparison uses one, four, or eight layers at width 128 with four heads. Only the learned-position models use positional embeddings.

B.2 How training advances

Training uses AdamW with learning rate 3×10^{-4}. Scale L uses 2×10^{-4}. Weight decay is .01 and the gradient clipping norm is 1.0. The batch size is 256, or 128 at scale L. The depth-eight and large-scale demand-one cells in Table 11 use learning rates 10^{-4} and 7×10^{-5}, respectively. Their counterparts at the default learning rate have ragged frontiers or late divergence.

Length curricula use stages with $n_{\max} \in \{4, 8, 16, 32\}$. Stage update limits range from 6k to 12k steps. After at least 500 steps, a stage advances when per-token accuracy reaches .98 or its optimizer-update limit is reached. Input lengths are sampled uniformly within each stage. Logarithmic training loop schedules use base count $T(n) = \lceil \log_2 n \rceil + 3$. Training jitters that count by ± 1; evaluation uses the base count. Proportional training loop schedules in this sweep use $T = \max\{2, \lceil mn \rceil\}$ with $m \in \{.25, .33, .5, .67, .75, 1, 1.5, 2, 3\}$. The fixed training loop count is $T = 12$. Streaming reveals token i from loop $i - 1$ onward and uses $T = n + 2$.

The horizon curriculum for chained A_5 and S_5 tasks supervises only the first h positions. The horizon increases from 4 to $n_{\max}$ within a stage. Operator-first S_5 training uses scale M. The first three stages have maximum lengths 2, 4 and 8. They receive 20k, 10k and 10k optimizer updates respectively. The remaining stages have maximum lengths 16 and 32. They receive 10k and 12k optimizer updates. Schedule-transfer experiments start from a trained source checkpoint. Continued training uses a maximum length of 16 for 6k optimizer updates. It then uses a maximum length of 32 for 8k optimizer updates. The paired loop-count continuation comparison in Table 7 instead uses 7,000 updates. The comparison across training settings and the schedule-transfer experiment use RTX 3090 hardware. Appendix A.3 lists the hardware for the fixed-update experiment.

B.3 Which models are evaluated

The frontier speed fit across different training settings has 23 measurable models among 27 attempted multiplier configurations. The broader Stage-2 grid contains 56 configurations. It varies groups and model scales, as well as seeds and streaming conditions. Each of the three association cohorts uses 16 seeds with the same training settings. Additional experiments compare tasks with matched group order. They also vary weight tying and positional encoding. Other comparisons vary the loop multiplier or streaming supervision. Further experiments change the curriculum or training loop schedule. The easy-to-hard comparison has eight models. The depth-width grid has 44 configurations including models trained at additional learning rates. Some cohorts share trained checkpoints.

The A_5 frontier cohort uses `a5v2` seeds 0 through 15. Five meet training-length exact-match accuracy $> .5$ and seven have a recorded frontier speed. The pooled scratch cohort combines `a5v2` seeds 0 through 15, 18, and 19 with `a5chain` seeds 0 through 7. It has $8/26$ training successes. Seeds 100 and 101 form a separate replication with $1/2$ successes. The pooled S_5 operator-first cohort combines the initial seeds 0 and 1 and a replication cohort with seeds 0 through 7. The two-model cohort supplies the frontier speeds in Table 5. Other frontier speed rows use consecutive seeds beginning at zero. The free-training medium-scale S_5 cohort succeeds at the training length in $3/4$ models. This free-training comparison uses different training settings from the operator-first curriculum. Each cohort counts each trained checkpoint once.

C What changes across tasks and models

C.1 Learning a local operation before a longer sequence

Cumulative group products separate local composition from repeated application. The operator-first curriculum first teaches the composition of two group elements, then extends the input to longer chains. The curriculum begins with pair composition and then increases chain length (Figure 12). Table 8 records the first logged step at which pair-composition accuracy reaches .98. These costs vary across operators and model scales. The compared groups differ in both their size and algebraic structure.

Table 8. Recorded optimization cost. First logged training step with per-token accuracy $\geq .98$ on pair composition (n=2). Costs differ across these operators; one scale up (M), the S_5 operator that takes 11k steps at S is learned in 6.5k steps.

	S_4 ($\lvert G\rvert$=24)	$\mathbb{Z}_{60}$	A_5 ($\lvert G\rvert$=60)	S_5 ($\lvert G\rvert$=120)
scale S	∼0.5k	2.5k	∼4k	11k
scale M	0.5k	2.0k	2.0k	6.5k

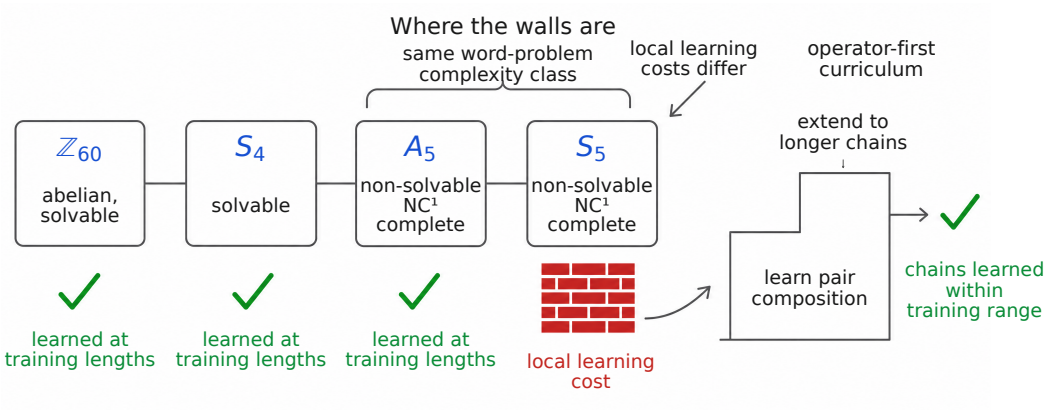

Figure 12. Learning a local operation and applying it along a chain. The group tasks differ in algebraic structure and local learning cost. The bracket refers to their fixed-group word problems. The curriculum emphasizes pair composition before extending to longer chains. Check marks indicate learning within the training range.

Operator-first S_5 training succeeds within the training range in the evaluated cohort. Performance at longer lengths and results from free-training and streaming comparisons are reported separately in Table 9. Each comparison uses the stated training settings. It includes all successes and failures within its cohort.

Table 9. Task and intervention comparisons. State replacement pools integer error counts. Each remaining comparison reports training success and longer-length performance.

Comparison	Result
State replacement	One A_5 and one S_4 checkpoint; 72 and 56 patch locations per evaluation batch, respectively; two batches per checkpoint; all no-change checks preserve predictions
A_5 length contrast	models successful at the training length have exact-match accuracy 0 at twice training length
S_5 operator-first	10/10 trained-length successes; exact-match accuracy $\leq$ .001 at twice training length
S_5 free-training replications	scale L ($\sim$ 13.26M) without PE and scale S with learned PE. Exact-match accuracy .00 for both variants, 2 seeds each
Streaming running parity	training-length exact-match accuracy $\geq$ .999, 8/8; at $4\times$ length. 0 through .998
Streaming, final-only	A_5. 2/2 failed; S_5. 8/8 failed; per-token $\leq$ .09
Streaming with per-loop supervision	A_5. 2/2 failed; S_5. 4/4 failed; per-token $\leq$ .09

C.2 When inputs appear and weights are shared

Streaming reveals input tokens on a fixed schedule and changes the attention mask accordingly. This gives a known reference for the timing measurement. Running parity learns the training-length task under this schedule. The tested non-solvable-group models fail to learn comparably accurate solutions, including when supervised at every loop on the revealed positions.

The tying-degree comparison holds effective depth at 12 and changes the number of distinct weight groups. Full tying reuses one group, partial tying cycles through several, and the untied setting gives each effective layer its own parameters. One comparison fixes width; the other reduces width as groups are added to keep the parameter count fixed. Table 10 reports them separately.

C.3 How depth and width affect frontier speed

Table 11 reports median frontier speeds over model-length pairs at $n \in \{16, 32\}$. A pair qualifies when its peak frontier coverage reaches .85, using per-position batch accuracy of at least .90

Table 10. Tying-degree comparison at fixed effective depth 12 (G weight groups cycled; mean per-token accuracy at n=64, 3 seeds per cell). With parameters matched, $G = 2$ outperforms full tying. The fully untied S_4 model has the lowest accuracy in this ladder. The A_5, $G \geq 3$ cells retain frontiers near the second position. Correct outputs at the first two positions and chance accuracy elsewhere predict $2/64 + (1 - 2/64)/60 \approx .0474$, about $2.84\times$ the chance floor.

		G=1 (tied)	G=2	G=3	G=4	G=6	G=12 (untied)
S_4	const. width	.21	.27	.30	.30	.32	.26
	param-matched	.21	**.28**	.27	.26	.25	**.15**
A_5	const. width	.10	.23	.17	.25	.23	.15
	param-matched	.10	**.15**	**.05**	.05	.05	.05

to define the frontier position. These thresholds apply within observation windows $H = 40, 64$, independently of the training loop count. Each depth or scale condition has two seeds. The demand-one eight-layer and large-model rows use separate lower-learning-rate cohorts. Their counterparts at the default learning rate pass $0/4$ cells. The one-layer demand-four cohort includes an extra lower-learning-rate condition.

Table 11. Frontier speed by block depth and width on S_4 (median v [range] over seeds and test lengths $n \in \{16, 32\}$; 2 seeds per cell plus comparisons at additional learning rates where noted). At demand four, the one-layer cohort has one eligible $n = 16$ cell and no eligible $n = 32$ cells; deeper blocks advance farther. The wider four-layer cohort has a lower median here but differs in other training settings. The marker n/a denotes cells outside these dedicated cohorts.

Block (width $\times$ layers)	demand 1	demand 4
128×1	0.98 [.93, 1.00]	$1/6$ cells pass; $n = 32$. $8/32$ to $13/32$
128×4	1.03 [1.00, 1.07]	4.67 [4.00, 5.33]
128×8	1.00 [1.00, 1.03]†	4.95 [4.00, 5.33]
256×2	1.00 [1.00, 1.03]	n/a
512×4	1.02 [1.00, 1.07]†	3.60 [3.11, 5.33]

†at lr 10^{-4} (128×8) and $7{\times}10^{-5}$ (512×4); at the default lr these cells train to high mean accuracy, but the frontier is ragged, or training diverges late. On A_5 at demand 1 the 128×4 cell fails to learn at every lr tried ($0/5$ models across 3 learning rates); the 128×1 and 128×8 cohorts succeed in $2/2$ and $1/2$ models, respectively. These counts reuse seed values across learning rates.

C.4 When extra loops help or hurt

Table 12 reports per-token accuracy over test-time loop counts. Each row averages the first three seeds in sorted order. Fewer seeds are used when fewer are available. Both successful and failed models are included. Values retain the three-decimal precision of the stored averages. Across tasks and model settings, extra loops can improve incomplete answers and later degrade them, with a different useful loop range in each setting.

D Results on the public running-sum benchmark

The easy-to-hard running-sum comparison uses the public data of Schwarzschild et al. (2021). We train on the 32-bit set and test on the 512-bit set. Training uses 10,000 examples. Each crop starts at the first input position and has a random length $n \in [4, 32]$. The model has no positional embeddings and uses scale s for 12k steps. Intermediate lengths use a generator matched to the task.

The successful models trained with proportional training loop schedules achieve high exact-match accuracy at four times the training length, with training success varying across seeds. All models trained with logarithmic training loop schedules fail at 512 bits. This benchmark measures accuracy extrapolation. Frontier speeds are unavailable. Accuracy declines between lengths 128 and 256. The convolutional Deep Thinking model exceeds 97% at 512 bits with its own architecture and training and evaluation settings (Bansal et al., 2022).

Table 12. Additional loops and overthinking. Mean per-token accuracy vs. evaluation loops t (saved three-decimal evaluation aggregates over the first three sorted seeds, or fewer if unavailable, without success filtering). The ETH row includes two failed models and one successful model. Cohorts trained with logarithmic loop schedules improve when t exceeds their training loop count and then degrade. Operator-first S_5 reaches near-perfect per-token accuracy at n=32. $\mathbb{Z}_{60}$ and models transferred to a different training loop schedule collapse sharply past their operating range.

		evaluation loops t						
Group	n	4	8	12	16	24	32	48
S_4 log S	32	0.382	0.688	0.947	0.969	0.907	0.846	0.780
S_4 log S	64	0.213	0.365	0.517	0.632	0.655	0.599	0.552
S_4 log L	32	0.384	0.788	0.957	0.859	0.741	0.686	0.644
S_5 op-first M	32	0.134	0.259	0.384	0.510	0.756	0.994	0.992
S_5 op-first M	64	0.071	0.134	0.197	0.258	0.382	0.507	0.640
transfer A_5 p→l	32	0.327	0.612	0.861	0.758	0.326	0.255	0.243
$\mathbb{Z}_{60}$ log S	32	0.348	0.748	0.858	0.741	0.284	0.124	0.082
ETH prop (mixed success)	32	0.579	0.633	0.715	0.730	0.756	0.760	0.769

Table 13 reports 512-example evaluations at longer lengths. The $n = 32$ values are the last logged training evaluations. A full 10,000-example reevaluation gives exact-match accuracy .9853 and .9585 at $n = 128$ for proportional seeds 2 and 3. It gives zero at $n = 512$ for all eight models. Other full-split values include .0017 at $n = 256$ for proportional seed 3 and .006 at $n = 64$ for logarithmic seed 0. The table and these full-split measurements use their respective sample sizes.

Table 13. Easy-to-hard running sums trained on the public 32-bit set. Exact-match accuracy by length and seed. The $n = 32$ column uses the final logged training evaluation; longer lengths use 512-example test evaluations. Full-split re-evaluation is summarized above.

Model condition	n=32	n=64	n=128	n=256	official 512
T=n, seed 2	1.00	1.00	1.00	0	0
T=n, seed 3	1.00	1.00	0.96	0	0
T=n, seed 0 (failed)	0.00	0	0	0	0
$T = n$, seed 1 (failed)	0.01	0	0	0	0
logarithmic, seed 0	1.00	0	0	0	0
logarithmic, seed 1	0.18	0	0	0	0
logarithmic, seed 2	0.05	0	0	0	0
logarithmic, seed 3	0.42	0	0	0	0
DT-Recall + prog. loss (Bansal et al., 2022)	reported $> 97\%$ at 512 bits (separate setting)				

E Related work

How looped models compute. Universal Transformers (Dehghani et al., 2019) tie weights across depth. Other architectures share parameters (Lan et al., 2020; Hutchins et al., 2022; Csordás et al., 2022; Tan et al., 2023; Csordás et al., 2024; Bae et al., 2025). Open models scale this idea (Huginn-3.5B, Geiping et al., 2025; Ouro, Zhu et al., 2025). Latent chain-of-thought moves reasoning into activations (Hao et al., 2024), while small recursive networks solve puzzles by iteration (Wang et al., 2025; Jolicoeur-Martineau, 2025). Looped transformers can be programmed to simulate graph algorithms (Giannou et al., 2023; Back de Luca & Fountoulakis, 2024). Their reasoning performance can rival deep non-looped models (Saunshi et al., 2025). Constructions length-generalize on n-RASP-L tasks (Weiss et al., 2021; Zhou et al., 2024; Fan et al., 2025) and emulate iterative solvers (Yang et al., 2024; Gatmiry et al., 2024). Blayney et al. (2026) study state convergence and attention stabilization in Ouro and Huginn. Huginn approaches a fixed point while Ouro need not. These observations distinguish the computation that produces a state from its behavior after settling. Lens studies find limited latent-CoT structure in Huginn (Lu et al., 2025). Our per-loop decoding applies logit-lens readouts (nostalgebraist, 2020; Belrose et al., 2023) after each shared update. Kohli et al. (2026) replace states at selected positions and iterations in small recurrent-depth models to locate retrieval and composition. We measure when outputs settle and replace intermediate states on cumulative group-product tasks. Activation patching supports circuit discovery in fixed-depth models

(Vig et al., 2020; Geiger et al., 2021; Meng et al., 2022; Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023; Yao et al., 2024); we apply it along loops. Feature-level tracing (Ameisen et al., 2025) and compiled ground-truth circuits (Lindner et al., 2023) are complementary. When states and local Jacobians are identical along a fixed-point tail, deterministic attribution based on them is also unchanged. Earlier states expose the computation leading to that tail. Path independence, or convergence to fixed points determined by the input, predicts generalization to greater depth in deep equilibrium models (Bai et al., 2019; Anil et al., 2022a). The recall architecture with progressive loss promotes convergence to input-determined fixed points (Bansal et al., 2022). Our comparisons measure similarity at $n = 16$ and track how training loop counts shape progress.

Measuring and perturbing intermediate states. Decoding identifies when information becomes readable. Perturbations test evolving states' causal roles and recovery (Churchland et al., 2012; Mante et al., 2013; Li et al., 2016). Our measurements follow this experimental logic. τ measures when decoded outputs settle, and state replacement tests whether intermediate states affect later answers.

Which computations can these models express? Under the cited bounded-precision and architecture assumptions, fixed-depth transformers sit in TC^0 (Merrill et al., 2022; Merrill & Sabharwal, 2023); Strobl et al. (2024) surveys these results. Solvable-group automata admit learned constant-depth shortcuts. Non-solvable groups (S_5, A_5) admit none unless $\mathrm{TC}^0 = \mathrm{NC}^1$ (Liu et al., 2023; Barrington, 1989). The backdrop is Krohn-Rhodes theory and the fine structure of NC^1 (Krohn & Rhodes, 1965; Barrington & Thérien, 1988). Log-depth looped-transformer constructions can use associative scans (Merrill & Sabharwal, 2025; Sanford et al., 2024); the analyzed fixed-depth SSMs fail S_5 state tracking under their computation assumptions (Merrill et al., 2024). These results describe the computations available for each algebraic structure. Our experiments measure the cost of learning the corresponding tasks.

A constant-depth self-attention construction computes parity with position-dependent features (Chiang & Cholak, 2022). It overcomes the soft-attention limitation of Hahn (2020). Learning to generalize parity to longer inputs remains hard (Bhattamishra et al., 2020; Delétang et al., 2023; Hahn & Rofin, 2024). Our experiments examine which computations training selects within this architectural capacity. Length generalization in standard transformers depends on positional encoding and input format (Anil et al., 2022b; Kazemnejad et al., 2023; Lee et al., 2024; McLeish et al., 2024). Scratchpads externalize intermediate state (Nye et al., 2021; Wei et al., 2022; Dziri et al., 2023). Chain of thought extends constant-depth models through serial token-space computation (Feng et al., 2023; Merrill & Sabharwal, 2024; Li et al., 2024; Goyal et al., 2024; Pfau et al., 2024). Loops add computation through repeated state updates without adding parameters. Halting and allocation policies determine its compute budget (Elbayad et al., 2020; Schuster et al., 2022; Snell et al., 2025). Iteration-wise supervision aligns loops with CoT steps (Yu et al., 2026). CLRS-style hints align processor iterations with algorithm steps (Veličković et al., 2022; Ibarz et al., 2022), following the Neural GPU (Kaiser & Sutskever, 2016). Hint-free models can instead learn parallel strategies (Rodionov & Prokhorenkova, 2023). Our streaming condition imposes an input-release schedule. The free-input conditions expose the entire input from the start.

Learning group operations. Grokked networks implement Fourier or coset circuits for modular and permutation arithmetic (Power et al., 2022; Nanda et al., 2023; Zhong et al., 2023; Chughtai et al., 2023; Stander et al., 2024). Induction heads are sequence-processing primitives (Elhage et al., 2021; Olsson et al., 2022). Fixed-depth reasoners perform depth-bounded parallel search, caching intermediate states at token positions (Brinkmann et al., 2024; Wang et al., 2024; Cabannes et al., 2024). Our focus is the repeated composition of a learned local operator $g \circ h$ along a sequence.

F What fixed-point measurements establish

An exact fixed point is a state left unchanged by another shared update. The following propositions characterize measurements on a stationary tail and deviations on an approximately stationary trajectory. Here H is the observation window in loops.

Proposition 1 (Exact form)**.** Let G be the autonomous update from $\mathbb{R}^{N\times D}$ to $\mathbb{R}^{N\times D}$ $G(h) = F_\theta(h, e)$ for fixed input embedding e, N token positions and hidden width D. Suppose G is differentiable at a fixed point $h^\star$. Assume the trajectory satisfies $h^t = h^\star$ for all $t \geq \tau^\star$. The readout is fixed and deterministic. Then the following properties hold.

(i) For $j \geq \tau^\star$ and $0 \leq k \leq H - j$, replacing $G^{H-j}(h^j)$ by $G^{H-j-k}(h^j)$ leaves the final state and decoded output unchanged.

(ii) The local Jacobian $J_t = DG(h^t)$ equals $J_{\tau^\star}$ for every $t \geq \tau^\star$. Any functional of these local Jacobians has the same value on identical arguments.

(iii) Let E be a fixed deterministic feature encoder. An attribution rule using only $E(h^t)$ and the local linearization at h^t has the same value throughout the tail if it has no explicit loop-index or remaining-horizon dependence.

These equalities alone do not identify the preceding computation in general. The Jacobian still constrains local first-order behavior.

Proof. The fixed-point identity gives $G^m(h^\star) = h^\star$ for every $m \geq 0$. Both trajectories in the first statement therefore end at $h^\star$. The second and third statements evaluate deterministic functions at the same state and local derivative. They contain no information that distinguishes different paths into that state. □

Proposition 2 (Approximate form)**.** Let $0 \leq t_0 \leq H$ be an integer. Suppose G is L-Lipschitz with $L \geq 0$ on a set containing $h^{t_0}, \ldots, h^H$. Assume $\|h^{r+1} - h^r\| \leq \epsilon$ for every $r \in \{t_0, \ldots, H-1\}$. For integers $t_0 \leq j \leq H$ and $0 \leq k \leq H - j$, omitting k updates yields

$$\|G^{H-j}(h^j) - G^{H-j-k}(h^j)\| \leq L^{H-j-k}\|G^k(h^j) - h^j\| \leq k\epsilon L^{H-j-k}.$$

If $L < 1$, the sharper bound is $\epsilon L^{H-j-k}(1-L^k)/(1-L)$. Under the same $L < 1$ condition, suppose G is differentiable. Assume also that its Jacobian is M-Lipschitz on this set. For $t_0 \leq t \leq s \leq H$, it then follows that $\|J_s - J_t\| \leq M\epsilon(1 - L^{s-t})/(1 - L)$.

Proof. Apply Lipschitz continuity to the common $H-j-k$ remaining updates. Telescope $G^k(h^j) - h^j$ over the omitted updates and use the bound ϵ on each increment. When $L < 1$, successive increment norms are at most $\epsilon, L\epsilon, \ldots, L^{k-1}\epsilon$. Their geometric sum gives the sharper state bound. The same argument yields $\|h^s - h^t\| \leq \epsilon(1-L^{s-t})/(1-L)$. Applying the Jacobian's M-Lipschitz condition proves the last inequality. □

The state bounds extend to decoded labels when the readout margin is large enough to preserve the predicted class. Bounds for continuous attribution statistics use their continuity assumptions. The sharper bound assumes contraction, which must be assessed separately from the size of observed updates.

G LIMITATIONS

Our experiments cover selected model sizes and synthetic sequence tasks. Larger models, other recurrent architectures, and pretrained language models remain to be evaluated. Testing these models on broader tasks, including natural-language reasoning, would establish how widely the relation between training loop count and frontier speed extends.