Where to Look First in Grokking: Attention and MLP Coherence in Modular Arithmetic

Ruian Lei[1] **Ruoxi Jiang**[2,3] **Marley M. B. R. Vellasco**[4]
M. Tanveer[5] **Zenglin Xu**[2,3]*
[1]University of Electronic Science and Technology of China
[2]Fudan University [3]Shanghai Academy of AI for Science
[4]Pontifical Catholic University of Rio de Janeiro [5]Indian Institute of Technology Indore
ryanlei5313@gmail.com roxie_jiang@fudan.edu.cn
marley@ele.puc-rio.br mtanveer@iiti.ac.in zenglinxu@fudan.edu.cn

Abstract

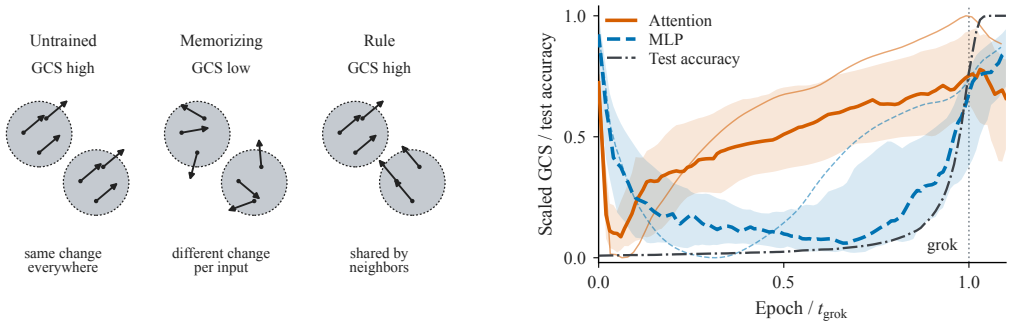
Networks trained on modular arithmetic memorize their training set long before learning a rule that generalizes to unseen inputs. Progress measures built from a known algorithm track its formation, but they need the algorithm in advance. We ask where in the network, and when in training, this formation can be seen without the algorithm. We introduce the Geometric Coherence Score (GCS), which retrospectively compares the local responses of attention and MLP outputs at inputs that the final model treats as neighbors. A shared rule can give similar responses there, whereas separate lookup entries need not. We find that across 104 grokking runs of one- to three-layer Transformers, MLP coherence falls to a trough and partly recovers by grokking in 103. Attention coherence rises near grokking and dips afterwards in 94. The trajectories differ most clearly in one-layer networks. To relate these trajectories to prediction, we reset the final model's attention or MLP to a pre-grokking checkpoint selected by attention's operand-swap symmetry. In one-layer addition and multiplication models, the attention reset costs more test accuracy than the MLP reset (mean drops 0.40 and 0.03), and more than any size-matched reset of random MLP units. The same MLP weights, held fixed during further training from the preceding coherence trough, let 15 of 17 runs reach 99% test accuracy within budget, against 4 with the untrained MLP. To focus further analysis on specific training stages, we also select short windows of consecutive checkpoints using attention coherence alone. These windows capture larger accuracy gains and attention-symmetry changes than random windows do and often differ from those selected by distance to the final model. These results point to MLP response changes before grokking and later changes in attention as starting points for investigating how a generalizing rule forms.

1 Introduction

How does a neural network develop an algorithm during training? In grokking, a network fits its training set long before it learns a rule that generalizes to unseen input pairs [1]. By a *rule*, we mean a learned procedure that solves the task across input pairs, including pairs not seen during training. Mechanistic analyses have identified such procedures, including Clock and Pizza, different algorithms for modular addition [2–5]. Progress measures built from a known algorithm trace its formation through memorization, circuit formation and cleanup [2]. Such measures need the algorithm in advance, and networks trained on the same task can implement different ones [3]. A measure that does not need the algorithm would let the same stages be followed in networks whose algorithm has not been reconstructed.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) GCS intuition

(b) Coherence trajectories

Figure 1: **The score and its trajectories.** (a) Dots are inputs, shaded groups are neighbors in the final model, and arrows are local output responses. (b) Median and interquartile range (IQR) over 35 one-layer runs, with one run (GELU addition, seed 777, selected post hoc) in thin lines. MLP coherence initially falls while test accuracy remains nearly flat. Each coherence curve is smoothed over three checkpoints and scaled to its own range up to 5,000 epochs after grokking. Test accuracy is not rescaled.

Without first reconstructing the algorithm, we ask whether attention and MLP responses change at the same stages of training. Test accuracy reports the network's success but does not show where in the network, or when in training, to look for these changes. We start by comparing how a component responds to local input changes at neighboring inputs, and we call the agreement of these responses coherence (Section 2).

The Geometric Coherence Score (GCS) measures it for attention and MLP outputs. We expect that if the network has a shared rule, neighboring inputs should get similar local responses, and that if it stores separate lookup entries, nothing forces them to. GCS compares input derivatives along probe directions. The final model fixes the token embeddings, neighborhoods and probes, so the measurement is retrospective and conditioned on a trained solution, without requiring an explicit algorithm or frequency choice. Other algorithm-free measures track function complexity, information dynamics or routing patterns [6–9]. We measure instead how one component's local responses agree across neighboring inputs.

We find that MLP and attention coherence change at different stages, most clearly in one-layer networks. In 103 of 104 grokking runs of one- to three-layer Transformers, MLP coherence drops to a trough and climbs part of the way back by grokking, and in 94 of them attention coherence peaks near grokking and dips afterwards (Figure 1b and Table 1). This pattern is consistent with the MLP first giving neighboring inputs dissimilar responses, as separate lookup entries would, and then shared responses as the network generalizes. In one-layer networks, replacing each run's grokking time with a random checkpoint lowers the count showing both patterns from 30/35 to a mean of 10.3 (a null comparison added post hoc).

The different trajectories lead us to ask how well the final network can work with each component's pre-grokking weights. We keep the rest of the final model fixed and reset either attention or the MLP to the same earlier checkpoint t^-, then measure accuracy without further training. We choose t^- just before the largest pre-grokking drop in the distance between attention outputs for the two operand orders. In one-layer addition and multiplication models, the attention reset costs a mean 0.40 of test accuracy and the MLP reset 0.03. Controls that match the number of replaced parameters, or replace activations instead of weights, give the same ordering (Section 4).

We then ask whether these MLP weights can also support continued learning. From the coherence trough, 15/17 runs reach 99% test accuracy within budget with the t^- MLP fixed, but only 3/17 with the trough MLP and 4/17 with the untrained MLP from the same run's initialization. From t^-, every run reaches the threshold both when the untrained MLP is fixed and when attention is fixed, although resetting attention is costly in the finished network.

To focus further analysis on specific training stages, we use attention coherence alone to select short windows of consecutive checkpoints. In held-out runs, a fixed selection rule gives coherence windows that rank above random windows for accuracy rises and for attention outputs becoming more similar under operand swaps (*symmetry events*). Distance to the final model ranks accuracy rises higher and

often selects different checkpoints. In the reset experiments, choosing the earlier weights from these windows preserves the ordering in 16/17 runs (Section 4).

The study makes three contributions:

1. *We introduce GCS, which compares local responses across neighboring inputs in a fixed final-model reference without specifying the learned algorithm. We use attention coherence to select checkpoint windows that rank above random windows for accuracy rises and symmetry events (Sections 2 and 5).*
2. *We find that MLP coherence falls and partly recovers by grokking, while attention coherence peaks near grokking and then dips, with the clearest distinction at one layer (Section 3).*
3. *In one-layer addition and multiplication runs, we show that resetting the final model's attention to t^- costs more test accuracy than resetting its MLP. Holding the MLP weights from t^- fixed also lets most networks learn from the coherence trough within budget (Section 4).*

Code and reproduction instructions are at `https://github.com/ryannL057/grokking-gcs`.

2 The Geometric Coherence Score

2.1 Input derivatives of attention and MLP outputs

At each input, we compute the local rate of change of attention, MLP and full-block outputs as the prediction-position embedding moves along a few directions. Jacobian-vector products (JVPs) evaluate these directional derivatives by automatic differentiation, without adding a finite perturbation to the input. We compare these responses between neighboring inputs, ignoring their sign. The score summarizes how well they align across neighbors, for the reference and probes we choose below. A high score alone does not show that a rule has been learned, since MLP coherence is already high at initialization (Figure 1a,b). We therefore follow changes in GCS within each run.

To compare checkpoints, we fix the sampled inputs and use the final model to define their embeddings, neighbors and probe directions. Every checkpoint is then probed at the same points and directions, where the finished network computes its answer, and only its own weights change the responses.

For modular addition, a learned procedure can treat different input pairs with the same sum alike, which motivates comparing responses across inputs. We obtain the comparison pairs from the model's representations without specifying a Fourier circuit. The reference can nevertheless carry answer information. In a check using all 3,830 training inputs from three seed-777 addition runs, every final-model neighbor shares the input's answer, against a random baseline of 0.0088 (Appendix B.5). This check uses a larger input set than the $N = 200$ GCS sample.

At the "=" position, attention has already made the output depend on both operands. We differentiate with respect to the embedding at this position while keeping the operands fixed, and compare the resulting local transformations between neighboring inputs. Moving the probe to the operands changes the measured trajectories (Section 3.3).

2.2 Choosing neighboring inputs and directions with the final model

Which inputs a rule treats alike is unknown in advance, so we let the finished network of the same run say which inputs are neighbors. Let f_{ref} be the final checkpoint from the same run and τ the prediction position in an input sequence $\mathbf{x}(a, b) = [a, b, =]$. At this position the input token is always "=", so we use the hidden state after the Transformer block, which already incorporates the operands. For each of N sampled training inputs $\mathbf{x}_i$, this gives $\mathbf{r}_i = h^{\mathrm{blk}}_{f_{\mathrm{ref}},\tau}(\mathbf{x}_i) \in \mathbb{R}^D$, where D is the model width. Connecting each state to its k nearest neighbors gives the graph of input pairs to compare.

To choose probe directions from the variation among neighboring representations, we stack the displacements from each representation to its neighbors $\mathcal{N}_i$ as rows,

$$\mathbf{M}_i = [\mathbf{r}_j - \mathbf{r}_i]_{j \in \mathcal{N}_i}. \tag{1}$$

The local probes are the top-d right singular vectors $\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,d}$ of $\mathbf{M}_i$. We give each input its own probe set, held fixed across checkpoints, and match probes at two inputs by singular-value rank. We

apply these output-state directions directly in the input embedding coordinates, without transporting them through the block.

We evaluate local responses at the attention, MLP and full-block outputs. Let $\mathbf{s} = \mathbf{e} + \mathbf{p}_t$ combine token embeddings $\mathbf{e}$ with checkpoint t's positional embeddings. The one-layer *path* maps are

$$F_{t,\mathrm{attn},\tau}(\mathbf{e}) = \mathrm{Attn}_t(\mathbf{s})_\tau, \qquad \text{attention output,} \tag{2}$$

$$F_{t,\mathrm{mlp},\tau}(\mathbf{e}) = \mathrm{MLP}_t(\mathbf{s} + \mathrm{Attn}_t(\mathbf{s}))_\tau, \qquad \text{MLP output,} \tag{3}$$

$$F_{t,\mathrm{overall},\tau}(\mathbf{e}) = \mathbf{s}_\tau + F_{t,\mathrm{attn},\tau}(\mathbf{e}) + F_{t,\mathrm{mlp},\tau}(\mathbf{e}), \qquad \text{full-block output.} \tag{4}$$

The MLP map includes upstream attention through the chain rule, so it measures the computation ending at the MLP output. For deeper networks we compose earlier blocks from checkpoint t into a prefix and measure the last block, with JVPs still taken with respect to the token embeddings.

2.3 Agreement across neighbors: one score per checkpoint

The response vector $\mathbf{z}_{i,r}^{t,\pi} \in \mathbb{R}^D$ gives the local rate and direction of output change for an infinitesimal change of the prediction-position embedding along $\mathbf{u}_{i,r}$. For path $\pi \in \{\mathrm{attn}, \mathrm{mlp}, \mathrm{overall}\}$ at checkpoint t, we compute it as a JVP at the fixed reference embedding $\mathbf{e}_i = \mathrm{Embed}_{f_{\mathrm{ref}}}(\mathbf{x}_i)$:

$$\mathbf{z}_{i,r}^{t,\pi} = \left.\frac{\partial F_{t,\pi,\tau}}{\partial \mathbf{e}_\tau}\right|_{\mathbf{e}=\mathbf{e}_i} \mathbf{u}_{i,r}. \tag{5}$$

For neighboring inputs (i, j), we compare these responses by their absolute cosine similarity. Writing $\widehat{\mathbf{z}} = \mathbf{z}/\max(\|\mathbf{z}\|, \epsilon)$ for a response normalized with a numerical floor, we average over matched probes:

$$\alpha_{ij}^{t,\pi} = \frac{1}{d}\sum_{r=1}^{d} |\langle \widehat{\mathbf{z}}_{i,r}^{t,\pi}, \widehat{\mathbf{z}}_{j,r}^{t,\pi} \rangle|. \tag{6}$$

The absolute value removes the sign ambiguity of singular vectors. To summarize agreement across the input set, we place the edge alignments in a weighted graph G, with $G_{ii} = 1$, $G_{ij} = \alpha_{ij}^{t,\pi}$ on symmetrized neighbor edges and zero elsewhere. We then count how many effective modes the spectrum of G has. For eigenvalues $\lambda_1, \ldots, \lambda_N$, we normalize their magnitudes and compute the spectral entropy:

$$q_m = \frac{|\lambda_m|}{\sum_n |\lambda_n|}, \qquad H(q) = -\sum_m q_m \log_2 q_m, \qquad \mathrm{GCS}(\mathbf{G}_{t,\pi}) = N - 2^{H(q)}. \tag{7}$$

As in the linear mapping number [6], which summarizes linear connectivity by spectral entropy, $2^{H(q)}$ is the effective number of spectral modes. When agreement links more of the graph, fewer modes typically arise and GCS is higher (Appendix B.2). The score lies between 0 and $N - 1$. If all edge alignments vanish, $\mathbf{G}$ is the identity and GCS is zero. We use absolute eigenvalues because the sparse matrix can have negative eigenvalues. Because GCS aggregates the spectrum, its ordering can differ from that of mean edge alignment (Appendix B.4). Repeating the calculation across checkpoints yields a coherence trajectory for each path (Algorithm 1, with implementation details in Appendix B.1).

3 Tracing attention and MLP coherence during grokking

3.1 Experimental setup

The primary experiment uses a one-layer decoder-only Transformer ($d_{\mathrm{model}} = 128$, 4 heads, $d_{\mathrm{mlp}} = 512$, no LayerNorm) trained with full-batch AdamW (learning rate 10^{-3}, weight decay 1.0) on $p = 113$ modular arithmetic with a 30%/70% train/test split. Each input is the sequence $[a, b, =]$, with the prediction read at the final "=" position. Addition, subtraction and multiplication use all p^2 ordered pairs, and division uses $a/b = ab^{-1} \bmod p$ for $b \neq 0$. The design spans 3 activations (ReLU/GELU/SiLU), 4 tasks and 3 seeds. A run groks when training accuracy reaches 99% before test accuracy, and test accuracy subsequently reaches 99%. We define t_{grok} as the epoch of steepest test-accuracy rise before that threshold. One SiLU addition run does not grok within its 15,000

epochs and is excluded only from summaries conditional on grokking. The trajectory summaries also include two- and three-layer replications measured at the last block, for 104 grokking runs in total. Two two-layer subtraction runs and one three-layer division run do not grok.

For the reset, patching and continuation tests we use the 17 one-layer addition and multiplication runs, with the earlier checkpoint t^- defined from attention-output symmetry in Section 4.1. Configurations and horizons (10,000–55,000 epochs) are in Appendix A. We compute GCS with $N = 200$ sampled inputs, $k = 15$ neighbors and $d = 8$ directions at checkpoints usually 200 epochs apart.

3.2 Coherence trajectories: a trough in MLP coherence before grokking

We first ask when GCS changes in each component. We find that MLP coherence falls and partly recovers by grokking in 103/104 runs. Attention coherence rises near t_{grok} and then dips in 94/104 runs. Together they form the joint signature, found in 93/104 runs (Table 1). Our reading, following the motivation of Section 2.1, is that neighboring inputs elicit less similar MLP responses during memorization and more similar responses as generalization develops. Section 3.3 tests how far this reading depends on the reference and probes. However, attention coherence also falls and recovers in 98/104 runs, and MLP coherence rises and falls near grokking in 29/35 three-layer runs but in only 9/35 one-layer runs. The components therefore differ most clearly at one layer. In Figure 2, attention falls and recovers at all three depths. Its trough broadens in the deeper networks, while the MLP more often shows a rise and fall near grokking, so the two components' broad shapes become more alike.

The fractions of runs meeting the MLP and attention criteria are 0.990 [0.971, 1.000] and 0.904 [0.851, 0.952], with bootstrap intervals over task, activation and seed clusters. Leaving out any one depth, task, activation or seed keeps these fractions at least 0.985 and 0.882 (Appendix C.1). In representative one-layer runs, the attention peak and MLP trough also persist when N, k or d is varied (Appendix B.6).

We also use a separate, horizon-based rule that defines early recovery as the smoothed, range-scaled MLP curve reaching 0.75 after its minimum and within the first 70% of the saved run. None of six runs ending below 50% test accuracy meets this rule, although four reach their attention-coherence peak late, after 45% of the run. Three of the six use weight decay 0.1. These counts use a different recovery threshold and time anchor from the by-grok counts above (Appendix C.3).

The timing is tied to grokking most clearly at one layer, where a post hoc null that replaces t_{grok} with random checkpoints lowers the joint count from 30/35 to a mean of 10.3. At two and three layers, the null means are close to the observed counts (Table 1). The pooled MLP count exceeds the 95th percentile under random anchors, surrogate curves and checkpoint-order shuffles. The joint count exceeds that threshold for anchors and surrogates but not for shuffles, while the attention count exceeds it only for random anchors (Appendix C.2).

Table 1: **Attention and MLP trajectory counts across model depths.** Counts cover 104 runs that grok. Rise–fall: changes of at least 5% of the saved range around the maximum in $[0.9, 1.1]\, t_{\mathrm{grok}}$, with no time limit on the later dip. Fall-and-recovery: both changes by grokking, each 10% of the range. Both criteria apply to both paths. The joint signature requires attention rise–fall and MLP fall-and-recovery. The null replaces t_{grok} with a random saved checkpoint (mean of 1,000 draws).

Diagnostic	1L (n=35)	2L last (n=34)	3L last (n=35)
Attention rise–fall	31/35	30/34	33/35
MLP rise–fall	9/35	22/34	29/35
Attention fall-and-recovery	35/35	32/34	31/35
MLP fall-and-recovery	34/35	34/34	35/35
Joint signature (random-anchor null mean)	30/35 (10.3)	30/34 (26.4)	33/35 (28.7)

3.3 Dependence on the reference and probes

Rescoring each one-layer run with a checkpoint near $0.5t_{\mathrm{grok}}$ or near its first 99% test accuracy gives the joint signature in 34/35 and 28/35 runs, respectively, compared with 30/35 under the final reference (Table 2). The intersection across all three own-run references is 23/35. A final reference from another seed of the same task gives the joint signature in 33/68 target/reference pairs. A reference therefore transfers only partly between independently trained models. Appendix B.5 shows the trajectories and additional reference controls.

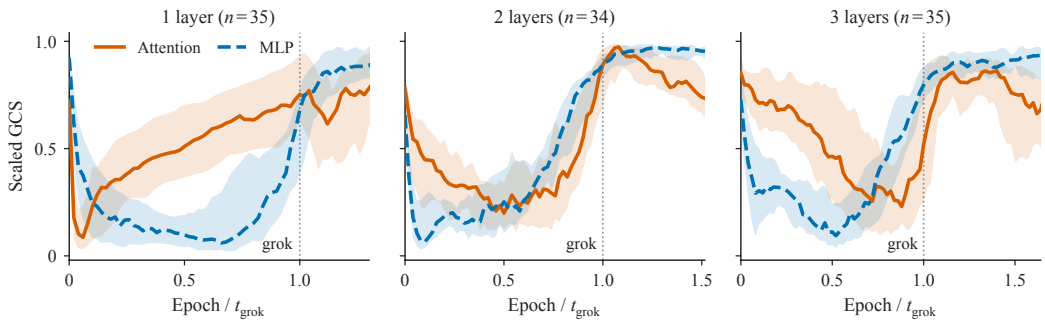


Figure 2: **Attention and MLP trajectories across depths.** Attention (solid) and MLP (dashed) coherence, median and IQR over 35 one-layer, 34 two-layer and 35 three-layer runs, measuring the last block. Epochs are divided by each run's t_{grok} and each score is scaled to its own range. The dotted line marks t_{grok}.

Table 2: **Trajectory counts under different references.** Counts give positive outcomes among the same 35 one-layer runs in the first four rows, and among 68 target/reference seed pairs in the last row.

Reference	Attention rise–fall	MLP fall-and-recovery	Joint
Own final	31/35	34/35	30/35
Own pre-grok	35/35	34/35	34/35
Own near-first-99%	28/35	35/35	28/35
All three own-run references	25/35	33/35	23/35
Other seed, same task	35/68	51/68	33/68

We find that matching probes differently can alter the timing conclusion even within a run. In an eight-run pilot using a best-fit (Procrustes) rotation instead of singular-value rank, the MLP's coherence still falls in all eight runs, but none recovers before grokking and six recover after it. Moving the probes from the prediction position to the operand positions can also reverse attention-trajectory correlations (Appendix B.3).

Two controls examine attention's role in the MLP trajectory. In one illustrative run with uniform (input-independent) attention weights and weight decay 0.1 rather than 1.0, attention coherence stays nearly flat while MLP coherence recovers (Appendix C.2). A JVP without attention's derivative leaves the MLP trajectory classifications unchanged in 35/35 runs (Appendix B.6). This control removes the backward chain through attention but keeps attention's forward input to the MLP.

4 Replacing and freezing attention and the MLP

We test the final model's dependence on each component's state by replacing its weights or activations with those from an earlier checkpoint, while keeping the rest of the network fixed. We then test continued learning with a fixed component at two training stages.

4.1 Choosing the earlier checkpoint

For addition and multiplication, exchanging the operands leaves the answer unchanged. Let $A_t(a, b)$ denote the attention output at the prediction position and define its operand-swap distance as

$$d_{\text{equiv}}(t) = \frac{\mathbb{E}_{a,b}\|A_t(a,b) - A_t(b,a)\|_2}{\mathbb{E}_{a,b}\|A_t(a,b)\|_2}, \tag{8}$$

with expectations over the valid swap grid. A value near zero means that the two operand orders give similar attention outputs. Across all 36 planned one-layer endpoints at $p = 113$, this distance is at most 0.026 for addition and multiplication and at least 1.17 for subtraction and division. A drop in this distance is the *symmetry event* used to evaluate checkpoint selection.

For each of the 17 grokking addition and multiplication runs, we choose t^- just before the sharpest decrease in d_{equiv} between saved checkpoints up to t_{grok}. Test accuracy enters only through t_{grok}, which ends the search. Section 5 evaluates a separate selection rule based on coherence. The chosen checkpoints have median test accuracy 58% and lie a median 200 epochs before t_{grok}. Every t^- falls

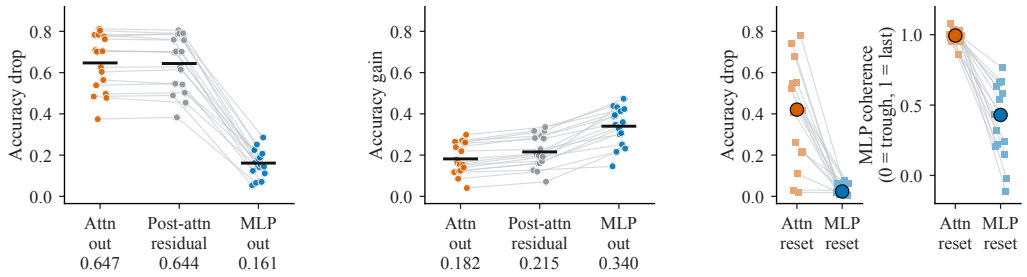
after the run's MLP-coherence trough, and coherence at t^- has recovered a median 0.48 of the way to its last scored value (range 0.02–0.73, post hoc).

In a cross-prime check at $p = 97$, a multiplication model reaches 99.85% accuracy while its operand-swap distance remains 1.0567. Its later computation produces output probabilities that are nearly invariant to operand order (Appendix D.1).

4.2 Weight resets and activation patching

We reset one component of the final model to its state at t^-, the donor checkpoint, holding the rest fixed [10, 11], and measure test accuracy without further training. We find that resetting attention costs more accuracy than resetting the MLP in all 17 runs, with mean drops 0.40 and 0.03 and medians 0.42 and 0.02 (Figure 3c). In two runs the two costs differ by less than 0.002. To compare resets of nearly equal size, we reset 255 of the MLP's 512 hidden units in 32 random subsets per run. Each subset has 65,535 parameters, compared with 65,536 in attention and 131,712 in the full MLP. The most damaging subset costs 0.025 on average over runs, and attention's cost exceeds every subset in all 17 runs. This includes the eight subsets per run chosen to approximate attention's donor-to-final weight displacement (Appendix D.2). The final model can therefore take its MLP weights from t^-, shortly before grokking, at little cost, while its predictions still depend on later changes in attention. Choosing the earlier weights from the start or center of the coherence window instead preserves this ordering in 16/17 runs (Section 5, Appendix D.1).

The dependence is uneven within attention. Its query, key, value and output matrices (Q, K, V and O) each contain 16,384 parameters, so they can be compared at equal reset size. Resetting Q alone has almost no effect. Resetting K or O, by contrast, costs more accuracy than resetting V in all 17 runs. Resetting V and O together costs 0.379 on average, compared with 0.148 for Q and K together (Table 15).



(a) Activation patching, t^- to final

(b) Activation patching, final to t^-

(c) Parameter resets into the final model

Figure 3: **Accuracy after activation patching and parameter resets.** (a,b) Activation patching at three sites, from t^- into the final model (a) and from the final model into t^- (b). Thin lines join the 17 addition and multiplication runs, and short horizontal lines mark means. (c) Accuracy drop when attention or the MLP of the final model is reset to its t^- state (17 runs), and post hoc MLP coherence after the reset, scaled to each run's trough (0) and last scored checkpoint (1). The coherence panel uses 16 runs, without ReLU addition seed 777, whose saved scores were not reproduced (Appendix D.3). Thin lines join each run's two resets, and large points are medians.

Patching t^- activations into the final model gives the same ordering without changing parameters. Replacing each input's attention output, post-attention residual or MLP output with its t^- activation lowers accuracy by a mean 0.647, 0.644 and 0.161. In the reverse direction, final activations raise the t^- network's accuracy by a mean 0.182, 0.215 and 0.340, most at the MLP output. That output is computed after the final attention and closes a median 84% of the accuracy gap (Figure 3a,b, Appendix D.2).

The ordering holds for donors chosen near the transition or on the memorization plateau. Donors nearest $0.8\,t_{\mathrm{grok}}$ keep it in 16/17 runs, and memorization-plateau donors keep it in 15/17. One donor per run drawn from a random window of the same width, a median 800 epochs after t_{grok}, keeps it in 10/17 (Appendix D.1). With further training, all 17 runs regain 99% test accuracy within 50 epochs after a reset (median 20 for attention and one for the MLP, Appendix D.2).

4.3 Continuing training with a fixed MLP

We next ask which MLP weights let the network continue learning from the MLP-coherence trough or from the same t^- used for resets. In the same 17 runs, we combine MLP weights from initialization, the trough or t^- with the remaining parameters from either stage. We hold the MLP fixed while the token and position embeddings, attention and unembedding continue training. We give all branches of a run the same additional budget B_i (2,000–14,800 updates), set by the original training time remaining after t^-. We count a branch as successful when it reaches 99% test accuracy within that budget (Appendix D.4).

At the trough, 15/17 runs reach the threshold with the t^- MLP fixed, compared with 3/17 for the trough MLP and 4/17 for the untrained MLP (Table 3). With nothing frozen, 12/17 reach the threshold (Appendix D.4). At t^-, however, all 17 reach the threshold with their untrained MLP fixed. The higher success count with the t^- MLP suggests that its changes after the coherence trough help an early network learn within the tested budget. In a post hoc comparison, the four trough networks that succeed with the untrained MLP have the shortest intervals to t^- (200–1,400 epochs, against at least 2,800 in the other runs). At t^-, their MLP coherence has also recovered the smallest fractions of its trough-to-last range (0.02–0.20, against at least 0.24 in the other runs). Tripling the budget raises success with the untrained MLP at the trough to 8/17.

Table 3: **Continued learning with a fixed MLP.** Counts reaching 99% within B_i. Columns give the starting stage of all remaining parameters, which continue training.

MLP fixed at	Remaining parameters at trough	At t^-
Untrained (initialization)	4/17	17/17
Coherence trough	3/17	15/17
t^-	15/17	16/17

We had predicted that attention would still have to change after t^-. Freezing it there instead, while the embeddings, MLP and unembedding train, succeeds in all 17 runs.

5 Selecting checkpoints using attention coherence

To test whether the attention-coherence curve alone locates when in training the network changes, we select windows of 11 consecutive checkpoints using only this curve, then evaluate the accuracy increase and operand-swap-distance decrease within each window. Distance to the final model falls as a checkpoint approaches the final state, while coherence rises as attention's local responses come to agree across inputs the final model treats as neighbors, so the two can select different checkpoints.

Our rule smooths attention GCS over three checkpoints and finds its minimum G_{trough} after initialization and before 45% of the trajectory. It then takes the first later local maximum that rises above a threshold set by the curve's scale and by its range after the minimum (Appendix C.4). If none qualifies, it uses the first crossing of 75% of that range, and otherwise the maximum after the minimum. The window spans $W = 11$ checkpoints centered on the chosen point. We fixed the rule and W, chosen from lengths 5–11, on seed 42 before any held-out evaluation. Only afterwards do we rank the accuracy increase or operand-swap-distance decrease within the window against those in all same-width windows. A high percentile means the selected window contains a larger change than most same-width windows in that run. Ties receive their upper rank, which is why a random window can score above 0.5 (Appendix C.4).

Across 23 held-out runs for accuracy and 11 addition and multiplication runs for symmetry (seeds 123 and 777), these coherence windows reach median percentiles of 0.830 and 0.934 (Table 4). Direct-J, which rebuilds the neighbor graph from each checkpoint's own embeddings and uses shared random probe directions, reaches 0.581 and 0.186 with the same rule, below the random-window baseline of 0.606 and 0.511. A largest-change rule added after that comparison raises it to 0.904 and 0.967. GrokAlign, a per-point Jacobian–input alignment score, reaches 0.473–0.574 and 0.011–0.341 over its four selection rules. Centered kernel alignment (CKA) [12] and attention-output L_2 distance to the final model reach 0.956 and 0.954 for accuracy.

The extended evaluation covers 71 held-out runs for accuracy and 35 for symmetry, including 48 new runs (four additional seeds, six in total). Without retuning, coherence windows reach median percentiles of 0.887 [0.830, 0.950] and 0.963 [0.934, 0.975], against 0.779 and 0.508 for random

Table 4: **Changes captured by selected windows on held-out runs.** GCS and Direct-J use the same peak rule in the first two rows. The third row uses a largest-change rule, added post hoc, for Direct-J's curve. Entries are median within-run percentiles of the accuracy increase or operand-swap-distance decrease in each window (23 runs for accuracy, 11 for symmetry). Higher percentiles indicate larger changes relative to other windows in the same run. GrokAlign [13], a per-point Jacobian–input alignment score, is summarized by its range over the four selection rules tested for it.

Diagnostic and selection rule	Acc. pct.	Sym. pct.
Attention GCS, post-trough peak	0.830	0.934
Direct-J, same peak rule	0.581	0.186
Direct-J, largest-change rule	0.904	0.967
Final-representation CKA [12]	0.956	0.908
Attention-output L_2 to final	0.954	0.964
GrokAlign, four-rule range	0.473–0.574	0.011–0.341
Random window	0.606	0.511

windows, while seven selectors by distance to the final model, using parameter, representation or component-output distances (Appendix C.4), reach 0.967–0.979 and 0.957–0.988. Coherence windows have median Jaccard overlap zero with the windows of four of these selectors, and the absolute differences between their starts have medians of 3,200–5,600 epochs (Appendix C.4).

When we change one of N, k and d at a time on seeds 123 and 777, median percentiles stay at 0.734–0.921 for accuracy and 0.778–0.934 for symmetry, although window centers move by up to a median 2,400 epochs and the symmetry intervals include chance at $k = 20$ and $N = 100$ (Appendix B.6). In the 11 held-out addition and multiplication runs, the coherence window is centered a median 200 epochs before t_{grok}, at median test accuracy 62.4%. Direct-J's change-based rules center their windows after grokking at near-final accuracy (Appendix C.4).

Contents of the selected windows. In one addition run (SiLU, seed 777), GCS and attention-output CKA select overlapping windows with similar accuracy gains (0.809 and 0.810), while the later L_2 window contains more change in attention-output symmetry (Table 12). Within the coherence window, attention outputs still differ under operand swaps while downstream logits are already more symmetric. Appendix C.5 reports these operand-swap distances and changes in attention-head patterns.

In five one-layer addition runs, we also compare attention coherence with restricted Fourier progress, the negative of the restricted loss of Nanda et al. [2]. This loss keeps only the $\cos(w(a+b))$ and $\sin(w(a+b))$ logit components of the key frequencies w and the constant term. Our implementation recomputes the key frequencies at each checkpoint instead of fixing them at the final model. Before grokking the two curves correlate in level, but their first differences correlate weakly (median 0.034), so the agreement largely reflects a shared training trend (Appendix C.6).

6 Related work

Hidden progress and component timing. Nanda et al. [2] track a modular-addition circuit before test loss improves, and Barak et al. [14] show hidden progress in parity learning. In Transformers that learn abruptly, attention maps can improve during the loss plateau [15]. ExPLAIND uses gradient path kernels to attribute predictions to parameters, training examples and training steps. Its grokking analysis identifies attention and the first MLP linear layer as a representation pipeline learned after memorization, with embeddings and output layers adapting around it [16]. GCS instead measures how one component's local responses agree across neighboring inputs in a fixed final-model reference.

Arithmetic circuits and learning dynamics. Fourier, Clock/Pizza, analytical-weight, numerical-integration and approximate-coset accounts identify the circuits [2–5, 17]. For two-layer MLPs on one-hot inputs, He et al. [18] derive single-frequency neurons, as at the max-margin solution [19, 20], with phases combining by majority vote and frequencies set largely at initialization. In ReLU MLPs, frequencies and phases read before generalization can define an accurate idealized network [21]. In sparse parity, a sparse subnetwork with growing norms comes to dominate prediction [22]. Theories explain grokking by circuit competition under weight decay [23], by representation learning and weight norm [24, 25], by a lazy-to-rich transition [26] and by feature emergence [27].

Diagnostics and interventions. GCS compares one path's input Jacobians between neighboring inputs. Its effective-mode count follows the linear mapping number, a spectral-entropy count of linear connectivity along input segments [6]. GrokAlign compares Jacobian and input directions at each input [13], and the average gradient outer product combines all inputs' gradients in one matrix [28].

Other diagnostics track representation geometry, from region migration and rank to intrinsic dimension and CKA similarity [7, 12, 29–32], and attention-head differentiation [33]. Gradient coherence and tangent-kernel analyses compare loss gradients or parameter Jacobians across examples or checkpoints [34–36], whereas GCS compares input Jacobians of one component's output in a fixed final-model reference. Symmetry, patching and resets examine function [10, 11, 37–41]. Optimizing only embeddings can also recover algorithmic behavior from random Transformers [42].

7 Discussion and conclusion

Implications. Without reconstructing the algorithm, GCS shows MLP and attention coherence changing at different stages of grokking in a fixed final-model reference, most clearly in one-layer networks. MLP coherence falls and partly recovers by grokking, and attention coherence rises near grokking. In the one-layer addition and multiplication runs, resetting attention to t^- costs more accuracy than resetting the MLP.

The continuation tests ask a different question from the resets, namely whether the rest of the network can continue learning with an earlier component held fixed. The t^- MLP supports learning from the coherence trough in more runs than either the trough or untrained MLP within the same budget. From t^-, the rest of the network learns with an untrained MLP held fixed or with attention held at its t^- state.

Progress measures built from a known algorithm trace how it begins to contribute to predictions [2], and GCS measures local response agreement without specifying the algorithm. In five addition runs, coherence and restricted Fourier progress share a training trend, but their step-to-step changes correlate weakly (median 0.034). For checkpoint selection, windows chosen by distance to the final model rank accuracy rises higher than coherence windows and symmetry events about as well, and the two kinds of window often select different checkpoints.

Limitations. The experiments use small modular-arithmetic Transformers without LayerNorm. LayerNorm pilots, sampled too coarsely for their fast grokking, leave the joint signature unresolved (joint signature in 2/11 one-layer and 0/10 two-layer runs, Appendix A.1). Same-protocol neural tangent kernel (NTK) or Fisher-alignment comparisons were not run. Deeper networks are measured only at their last block. Because GCS is retrospective, the final reference can miss precursor structure absent from the final model. Trajectories also depend on coordinates, probes and reference, since probes at the operand positions can reverse attention correlations and reference neighborhoods can encode answers. With Procrustes matching, none of eight pilot runs shows pre-grok MLP recovery (Section 3.3). This probe dependence limits what the recovery time tells us about rule formation. A detector that does not use the grokking time finds the joint signature in only 3/23 of the held-out one-layer runs of Section 5, and the horizon-based rule of Section 3.2 finds it in 2/24 trajectories cut off at 90–95% test accuracy (Appendix C.3). The reset tests cover the 17 addition and multiplication runs, where t^- is defined by task symmetry, and use one donor checkpoint per run for each donor rule. The resets measure accuracy right after a replacement, with the rest of the network fixed, and the reverse-patching and recovery results leave open when each component acquired its role. Frozen-component outcomes depend on the continuation budget and on which remaining parameters can adapt (Appendix D.4). Whether coherence windows make algorithm reconstruction easier, or expose circuit changes that windows chosen by distance or accuracy miss, remains open.

Acknowledgments and Disclosure of Funding

We thank Yiyi Ran for extensive help with figure design and preparation, and Xuejiao Li for helpful suggestions on improving the readability of the manuscript. Ruian and Zenglin acknowledge support from the National Key Research and Development Program of China under Grant 2018AAA0100204. Marley acknowledges support from the Brazilian agencies CNPq (Process 305962/2024-8) and FAPERJ (Process E-26/200.068/2026). Ruian thanks NeurIPS 2026 for financial assistance covering conference registration and accommodation. The authors declare no competing interests.

References

[1] 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.

[2] 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.

[3] 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, pages 27223–27250, 2023.

[4] Andrey Gromov. Grokking modular arithmetic. arXiv preprint arXiv:2301.02679, 2023.

[5] Gavin McCracken, Gabriela Moisescu-Pareja, Vincent Létourneau, Doina Precup, and Jonathan Love. Uncovering a universal abstract algorithm for modular addition in neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[6] Ziming Liu, Ziqian Zhong, and Max Tegmark. Grokking as simplification: A nonlinear complexity perspective. In *NeurIPS Workshop on Unifying Representations in Neural Models (UniReps)*, 2023.

[7] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep networks always grok and here is why. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20722–20745. PMLR, 2024.

[8] Kenzo Clauw, Sebastiano Stramaglia, and Daniele Marinazzo. Information-theoretic progress measures reveal grokking is an emergent phase transition. In *ICML Workshop on Mechanistic Interpretability*, 2024.

[9] Ziyue Li, Chenrui Fan, and Tianyi Zhou. Grokking in LLM pretraining? Monitor memorization-to-generalization without test. In *International Conference on Learning Representations*, 2026.

[10] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 23(67):1–28, 2022.

[11] Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. arXiv preprint arXiv:2404.15255, 2024.

[12] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 2019.

[13] Thomas Walker, Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. GrokAlign: Geometric characterisation and acceleration of grokking. arXiv preprint arXiv:2506.12284, 2025.

[14] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *Advances in Neural Information Processing Systems*, volume 35, pages 21750–21764, 2022.

[15] Pulkit Gopalani and Wei Hu. What happens during the loss plateau? Understanding abrupt learning in transformers. In *Advances in Neural Information Processing Systems*, volume 38, pages 54928–54960, 2025.

[16] Florian Eichin, Yupei Du, Philipp Mondorf, Maria Matveev, Barbara Plank, and Michael A. Hedderich. ExPLAIND: Unifying model, data, and training attribution to study model behavior. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 27688–27719. PMLR, 2026.

[17] Chun Hei Yip, Rajashree Agrawal, Lawrence Chan, and Jason Gross. Modular addition without black-boxes: Compressing explanations of MLPs that compute numerical integration. arXiv preprint arXiv:2412.03773, 2024.

[18] Jianliang He, Leda Wang, Siyu Chen, and Zhuoran Yang. On the mechanism and dynamics of modular addition: Fourier features, lottery ticket, and grokking. arXiv preprint arXiv:2602.16849, 2026.

[19] Depen Morwani, Benjamin L. Edelman, Costin-Andrei Oncescu, Rosie Zhao, and Sham Kakade. Feature emergence via margin maximization: Case studies in algebraic tasks. In *International Conference on Learning Representations*, 2024.

[20] 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 *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 523–531. PMLR, 2025.

[21] Anand Swaroop. Latent algorithmic structure precedes grokking: A mechanistic study of ReLU MLPs on modular arithmetic. arXiv preprint arXiv:2603.23784, 2026.

[22] William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. In *ICLR Workshop on Understanding Foundation Models*, 2023.

[23] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. arXiv preprint arXiv:2309.02390, 2023.

[24] Ziming Liu, Ouail Kitouni, Niklas S. Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 34651–34663, 2022.

[25] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *International Conference on Learning Representations*, 2023.

[26] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *International Conference on Learning Representations*, 2024.

[27] Yuandong Tian. Li_2: A framework on dynamics of feature emergence and delayed generalization. In *International Conference on Learning Representations*, 2026.

[28] Neil Rohit Mallinar, Daniel Beaglehole, Libin Zhu, Adityanarayanan Radhakrishnan, Parthe Pandit, and Mikhail Belkin. Emergence in non-neural models: Grokking modular arithmetic via average gradient outer product. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 42834–42856. PMLR, 2025.

[29] Thomas Walker, T. Mitchell Roddenberry, Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. The geometric structure of models learning sparse data. arXiv preprint arXiv:2605.08464, 2026.

[30] Silvia Bernardi, Marcus K. Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C. Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4): 954–967.e21, 2020. doi: 10.1016/j.cell.2020.09.031.

[31] Benjamin Matthias Ruppik, Julius von Rohrscheidt, Carel van Niekerk, Michael Heck, Renato Vukovic, Shutong Feng, Hsien-chin Lin, Nurul Lubis, Bastian Rieck, Marcus Zibrowius, and Milica Gasic. Less is more: Local intrinsic dimensions of contextual language models. In *Advances in Neural Information Processing Systems*, volume 38, pages 67795–67828, 2025.

[32] David Yunis, Kumar Kshitij Patel, Samuel Wheeler, Pedro Henrique Pamplona Savarese, Gal Vardi, Karen Livescu, Michael Maire, and Matthew Walter. Grokking, rank minimization and generalization in deep learning. In *ICML Workshop on Mechanistic Interpretability*, 2024.

[33] George Wang, Jesse Hoogland, Stan van Wingerden, Zach Furman, and Daniel Murfet. Differentiation and specialization of attention heads via the refined local learning coefficient. In *International Conference on Learning Representations*, 2025.

[34] Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*, 2020.

[35] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[36] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: An empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems*, volume 33, pages 5850–5861, 2020.

[37] Zhiquan Tan and Weiran Huang. Understanding grokking through a robustness viewpoint. arXiv preprint arXiv:2311.06597, 2024.

[38] Hyeonbin Hwang and Yeachan Park. Intrinsic task symmetry drives generalization in algorithmic tasks. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 49034–49077. PMLR, 2026.

[39] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023.

[40] Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pages 16318–16352, 2023.

[41] Niladri S. Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. In *International Conference on Learning Representations*, 2020.

[42] Ziqian Zhong and Jacob Andreas. Algorithmic capabilities of random transformers. In *Advances in Neural Information Processing Systems*, volume 37, pages 104357–104382, 2024.

A Experimental scope and reproducibility

Reading guide. Table 5 maps each evidence set to the appendix that reports its definitions, summaries and exceptions.

Table 5: **Evidence sets, their scope and where they are reported.** Counts refer to runs unless stated otherwise.

Evidence	Models and inclusion	Question addressed	Appendix
Primary trajectories	104 grok-observed runs; four tasks, three activations, 1–3 layers	Recurrence and timing of the attention and MLP trajectory features, their nulls and the order of upturns	C.1–C.2
Measurement checks	35 one-layer runs and smaller pilots	Aggregation, rank matching, probe position, references, geometry size and upstream gradients	B
Truncated and non-grokking runs	24 one-layer runs cut at 90–95% test accuracy; 7 runs that do not grok	Whether the signature appears before a solved endpoint or without generalization	C.3
Window selection	71 held-out one-layer runs across six seeds; 48 newly trained	Accuracy and symmetry changes with a rule selected on seed 42	C.4
Restricted Fourier progress	5 one-layer addition runs	Association with attention coherence after removing the training trend	C.6
Component interventions	17 one-layer add/mul runs; coherence readings on 16	Immediate replacements and continued training with a fixed component	D
Scope pilots	LayerNorm, S_5, C_{120} and D_{60} runs	Behavior outside the primary architecture and tasks	A.1

Training configuration. The primary decoder-only Transformers have width 128, four attention heads, MLP width 512, no LayerNorm, and ReLU, GELU or SiLU activation. Seeds are 42, 123 and 777. Full-batch AdamW uses learning rate 10^{-3}, $\beta = (0.9, 0.98)$, numerical $\epsilon_{\mathrm{Adam}} = 10^{-8}$ and weight decay 1.0. A seeded 30%/70% split partitions ordered operand pairs at $p = 113$. Addition, subtraction and multiplication use all p^2 pairs. Division excludes zero divisors. Each input is the sequence `[a,b,=]` and only the prediction-position output is read. One SiLU/addition one-layer run fails to grok and remains in the 36-run endpoint scope check, but not the 35-run grok-conditioned summaries. Table 6 gives training horizons and GCS sampling ranges.

All learned matrices start with independent zero-mean Gaussian entries. Token and position embeddings, attention matrices and both MLP matrices have variance $1/128$ (standard deviation $1/\sqrt{128}$), and the untied unembedding has variance $1/114$. MLP biases start at zero. Training and evaluation use float64. One epoch is one full-batch update. Accuracy is logged every 200 updates in the primary training configuration, before that iteration's update.

Non-grokking controls. Seven runs fit the training set without reaching 99% test accuracy (Appendix C.3). Four use the primary recipe. They are two two-layer subtraction runs (final test accuracy 4.0–4.3%), the one-layer SiLU addition run that does not grok within its 15,000 epochs (37.8%) and a three-layer division run that ends at 94.5%. The other three are one-layer SiLU addition runs trained with weight decay 0.1 (0.4–1.7%), and two of them are scored at nine checkpoints with $N = 80$ and $k = 10$. These controls are not fully matched to the primary runs for weight decay, depth or training horizon.

Table 6: **Recorded ranges for the 104 primary runs.** Training horizon is the final saved model epoch plus one. GCS count is the number of analyzed samples, including epoch zero. Ranges describe variation across runs with different evaluation budgets.

Cohort	Runs	Training horizon (epochs)	GCS samples per run
1L	35	10,000–55,000	50–275
2L last	34	15,000–50,000	31–250
3L last	35	12,000–40,000	60–200

Saved records and sampling. Epoch indices start at zero. Seven runs were trained beyond their initially configured horizon. GCS is sampled every 200 epochs in 100/104 runs, every 100 or 500 in the four exceptions. The final model is included in 14 series, and 87 end at the preceding regular

sample. Three three-layer addition series with seed 777 stop at epoch 20,000 although training continues to 29,999. Headline ranges and later-dip searches use only these saved samples.

Compute resources. Experiments used a single NVIDIA GeForce RTX 4090 (24 GB), without distributed training. Table 7 lists representative wall-clock times. Reference construction scales with geometry size (including pairwise distances), and path evaluation scales with the number of checkpoints and paths. Exploratory and failed runs add to the reported cohort costs. Cheaper approximations were not assessed.

Table 7: **Representative wall-clock times** (single RTX 4090).

Stage	Epochs	Wall time	Note
Training (1L, 20k ep.)	20,000	$\sim$19 min	full-batch AdamW
Reference geometry build	—	$\sim$2 min	N=200, k=15, d=8
GCS extraction (100 ckpts)	—	$\sim$28 min	3 paths, JVP-based evaluation
Total (training + GCS)	—	$\sim$49 min	per run

Study provenance. The original submission reported the primary trajectories with their threshold sweeps, cluster bootstrap and leave-one-factor-out check, the uniform-attention and detached-attention controls, most of the measurement and reference checks, and the non-grokking controls. It also reported checkpoint selection on seeds 123 and 777 with the GrokAlign comparison, the task-algebra checks, the commutative replacements with their submodule, matched-null, donor, patching and recovery checks, and the comparison with restricted Fourier progress. It read the level correlation between attention coherence and restricted Fourier progress as agreement, which the trend controls of Appendix C.6 revise. The rebuttal added the Direct-J comparison, the geometry sweep and six-seed extension of checkpoint selection, the truncated trajectories of Appendix C.3, the previous-layer and hybrid references and the S_5 pilots.

After the rebuttal we added corrections and qualifications of these results. These include the cross-path trajectory counts, the recorded horizons and sampling ranges, and the near-tie qualification for reset costs. The method description was corrected to use displacements from each reference point, checkpoint-specific positional embeddings and a floor on each response norm. Further checks are the null baselines of Table 10, the Procrustes counts at matched epochs, the uniform-weight sweep, the 35-run comparison of aggregations, the reference averaged over two blocks, the proximity baselines over six seeds, the ranking of selected donors by reset cost, the detector without grok timing, the Fourier trend controls with the note on per-checkpoint key frequencies, and the LayerNorm pilots.

Post-rebuttal analyses also include the patching gap-closure shares. Two further analyses were added after the rebuttal as new results, the order of the attention and MLP upturns by depth (Appendix C.2) and MLP coherence after the t^- resets (Appendix D.3). Both use runs we had already examined. The submitted timing-separation row (34/35, 33/34 and 34/35) was unsigned. Appendix C.2 reports the signed upturn order instead, which the main text does not use. The rebuttal's broad-timing counts under alternative references (35/35 and 67/68) concerned the full-block upturn relative to grokking, not the ordering of attention and MLP upturns. We do not use those counts as evidence for component ordering. Two submitted secondary claims are not repeated. The submission reported that the symmetry midpoint precedes the attention window maximum in all 17 commutative runs and attached an exact sign test. That test ignores that the maximum is searched only within $[0.9, 1.1]\, t_{\mathrm{grok}}$, which fixes the order in the 6 runs whose midpoint falls before $0.9\, t_{\mathrm{grok}}$, and pairing each midpoint with another run's maximum still orders all 17 with probability 0.20. The count is therefore descriptive. The submitted comparison with parameter-norm and participation-ratio selectors was not repeated in Table 4. The rebuttal also described multiplication and division as spectrally diffuse, with about 9.5% of Fourier energy in the five strongest components. That figure used residue order, the natural basis for addition but not for multiplication. In discrete-log order the final token embeddings of the nine one-layer multiplication runs place a median 96% of their nonconstant Fourier energy in five components and those of the nine division runs 99.8%, against 9.8% and 9.5% in residue order. The low figure therefore reflects the basis and does not argue against Fourier structure in these tasks.

The nulls for the upturn order were written down after the observed counts were known and before the nulls were computed. The coherence readings are descriptive and depart from their written protocol in one place, noted where they are reported. The two-stage frozen-component continuations

(Appendix D.4) were also conducted after the rebuttal, on the same 17 runs as the resets. Their branch, initialization-MLP and extended-budget protocols were written before those experiments, but after inspecting the original trajectories and reset results. No protocol was publicly registered. Reimplementations by the same analyst reproduce the saved arrays and metrics. They check numerical agreement on shared evidence, not replication on new data.

Availability. Code, training configurations, saved results and reproduction instructions are available at `https://github.com/ryannL057/grokking-gcs`. The repository documents data provenance and reproduction coverage.

Assistance and excluded evidence. LLM tools assisted with writing, implementation and protocol specification. Measurements come from saved checkpoints and model outputs. Authors are responsible for all claims and calculations. Two exploratory comparisons are excluded. An operand-swap augmentation reversed training pairs after the split and so exposed 28.9–29.8% of the evaluation inputs during training. The primary experiments and interventions use no augmentation. In 24 runs with doubled attention or MLP weight decay, mean speedups were 0.98 and 1.13 for commutative tasks against 1.35 and 1.45 for non-commutative tasks, so they did not support a commutativity-specific acceleration.

A.1 Scope pilots

LayerNorm pilots sample GCS every 1,000 epochs, although many of them grok within 1,200–3,600 epochs. Among eligible runs, strict joint signatures occur in 2/11 at one layer and 0/10 at two layers, and fall-and-recovery without grok-relative timing occurs in both paths in 10/11 and 10/10. The joint signature is unresolved at this resolution. In pilots on composition in the symmetric group S_5, whose training fractions differ across runs, 5/9 runs grok and 4/5 of these pass the joint signature. A matched cohort on the group operation of the cyclic group C_{120} and the dihedral group D_{60} has 9/9 grok-observed runs per group (20k/60k budgets). Attention rise–fall occurs in 7/9 and 9/9, and MLP recovery in 8/9 and 9/9. None of these pilots enters the 104-run counts or the intervention evidence.

B Measurement validity and reference sensitivity

B.1 What the fixed reference specifies

The reference supplies contextualized prediction-position block outputs r_i, neighbor indices $\mathcal{N}_i$, local SVD directions $u_{i,r}$, and token embeddings e_i. Checkpoint parameters supply the positional embeddings and the path map. We use directions chosen in the reference states directly as probes in the residual coordinates. Input sequences, their embeddings and graph positions remain paired by sample index throughout the main measurement.

Algorithm 1 Default spectral score for cross-input coherence in a fixed reference

Require: Reference f_{ref}, checkpoint f_t, inputs $\{\mathbf{x}_i\}_{i=1}^N$, neighbors k, directions d, path π, position τ

Compute reference block-output states $\mathbf{r}_i$ and embedded basepoints $\mathbf{e}_i = \mathrm{Embed}_{f_{\mathrm{ref}}}(\mathbf{x}_i)$
Find each state's k nearest neighbors in $\{\mathbf{r}_i\}$ and save the directed lists and their symmetrized edge union
for each input $\mathbf{x}_i$ **do**
    Subtract $\mathbf{r}_i$ from each of its k directed neighbors to form $\mathbf{M}_i$ and take the top-d right singular vectors $\{\mathbf{u}_{i,r}\}_{r=1}^d$
    Compute $\mathbf{z}_{i,r}^{t,\pi} = (\partial F_{t,\pi,\tau}/\partial \mathbf{e}_\tau)|_{\mathbf{e}_i}\, \mathbf{u}_{i,r}$ for each direction r
end for
Initialize $G = I_N$ (zero off-diagonal entries)
for each symmetrized neighbor edge (i, j) **do**
    Set $G_{ij} = G_{ji} = \frac{1}{d}\sum_{r=1}^d |\langle \widehat{\mathbf{z}}_{i,r}^{t,\pi}, \widehat{\mathbf{z}}_{j,r}^{t,\pi}\rangle|$, where $\widehat{\mathbf{z}} = \mathbf{z}/\max(\|\mathbf{z}\|, \epsilon)$
end for
Compute eigenvalues λ_m of G, then $q_m = |\lambda_m|/\sum_n |\lambda_n|$ and $H(q) = -\sum_m q_m \log_2 q_m$
return $\mathrm{GCS} = N - 2^{H(q)}$ for checkpoint t and path π

The MLP-path Jacobian is $Dm(h_i)(I + J_{\text{attn}}(e_i))$ in a one-layer model, so both its forward input and upstream derivative depend on the attention state. The residual identity cannot prevent a zero response when $Dm(h_i) = 0$. A zero response contributes zero alignment under the norm floor in Equation 6.

Implementation details. To sample 200 training inputs without replacement, we continue the run's seeded random stream. The saved input list fixes both the sampled inputs and their order, which the seed alone cannot recover. The k-NN search uses Euclidean distance on unnormalized reference states, with candidates restricted to these 200 inputs and self-neighbors excluded. Each SVD uses the original 15 directed neighbors. Symmetrization by edge union is used only when constructing the score matrix.

For depth comparisons, reference states are the final block's outputs at the prediction position. Each checkpoint adds its own positional embeddings to the fixed final-model token embeddings and runs its earlier blocks before the measured block. We compute JVPs by automatic differentiation at the unchanged token embeddings, with a separate direction tensor nonzero only at the prediction position.

Models, JVPs and score matrices use float64. The pairwise alignment uses a norm floor of 10^{-8}, and the batched and response-reconstruction computations use 10^{-12}. Both divide by the product of the two floored norms, as in Equation 6. The eigenspectrum is computed in float64, and entropy sums terms with normalized magnitude $q_m > 10^{-12}$. Saved graphs and probes are reused in reconstruction checks. The response-norm check below reports how far its sampled responses lie above either floor.

B.2 Aggregation, limiting cases and numerical checks

For a dense all-ones alignment matrix, one spectral mode carries all mass, so $H = 0$ and GCS is $N - 1$. For the identity, $H = \log_2 N$ and GCS is zero. The all-ones case is an ideal limit, not the upper endpoint of an arbitrary sparse k-NN graph. On a sparse graph, both edge weights and connectivity affect the absolute-eigenvalue spectrum, so increasing all edge weights need not increase GCS.

We checked how large a dip this could produce by sweeping uniform edge weights, $G(a) = I + aA$, over 1,001 values in $[0, 1]$ on all 35 one-layer reference graphs. Only 6/35 sweeps are nondecreasing throughout, but the largest drawdown is 2.98% of the swept score range. No sweep produces the 5%-range dip required by the attention criterion. Over each path's observed mean-edge range, all 35 MLP sweeps are nondecreasing. This rules out that uniform-weight artifact but does not establish monotonicity for heterogeneous weights.

A synthetic circular graph ($N = 120, k = 10, d = 8$) yields GCS 56.5 and mean edge alignment 0.982 when neighboring transformed bases share a smooth rotation, versus 10.7 and 0.144 for independently random transformed bases. On eight representative runs with nine checkpoints each and reduced $N = 80, k = 10, d = 8$, adjacency normalization that keeps the unit diagonal gives median attention-trajectory Spearman $\rho = 0.983$ and a maximum within one checkpoint in 7/8. Removing the diagonal gives $\rho = 0.692$ and 0/8. Keeping or removing the unit diagonal changes the score's absolute scale and trajectory.

Across 19,200 JVPs from four representative model regimes at three stages, none has norm below 10^{-6} (minimum 0.135).

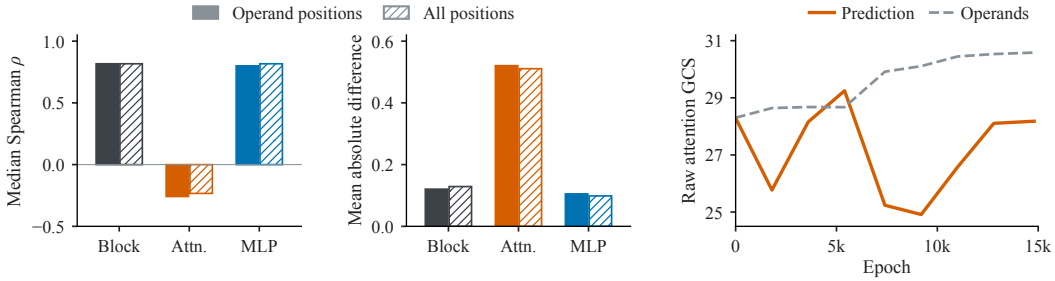
B.3 SVD rank matching and probe-position counterexamples

At $d = 8$, the ratio of the first to the last retained singular value across 600 neighborhoods from three addition models has mean $\sigma_1/\sigma_d = 5.80$ and minimum 2.85. Adjacent singular values can nevertheless have a ratio as small as 1.001, so global anisotropy does not ensure stable matching by singular-vector rank. In the eight-run reduced control, random rank derangements and random orthonormal probes lower attention-trajectory correlations to 0.383 and 0.283, with maxima within one checkpoint in 1/8 each.

Procrustes matching preserves the attention curve (median $\rho = 0.98$, maxima within one checkpoint in 8/8) but changes MLP timing (median $\rho = 0.04$). At matched sampled epochs, the pre-grok MLP fall-and-recovery criterion is met in 6/8 runs with singular-value rank matching and 0/8 with

Procrustes matching. Under Procrustes, the MLP curve falls in all eight runs and recovers after grokking in six.

Moving probes from prediction to operand positions, or to all positions, changes attention trajectories in a 12-run seed-42 control (four tasks, three activations; nine checkpoints; $N = 80, k = 10, d = 8$; Figure 4). Operand probes average the JVP responses from the two operand positions before computing GCS. All-position probes also include the prediction position. All three settings read the component output at the prediction position. Median correlations with the prediction-position curves are $-0.26/-0.23$ for attention, $0.80/0.82$ for MLP and $0.82/0.82$ for the full block. Operand and all-position attention probes are themselves nearly identical ($\rho = 1.00$).



(a) Correlation (b) Difference (c) Attention trajectory

Figure 4: **Changing the probed token position changes the measured response.** Twelve one-layer runs, nine checkpoints each, with $N = 80$, $k = 10$, $d = 8$. (a) Median correlation with prediction-position probes. (b) Mean absolute difference after range normalization, also relative to prediction-position probes. (c) Attention trajectories for ReLU addition, seed 42. All is omitted in (c) because it nearly overlaps Operands.

B.4 Alternative summaries of the same response graph

For the aggregation comparison, we use the same reference, probes and $N/k/d$ in 35 runs, with 50–101 saved checkpoints per run. GCS values at matched epochs have median correlation 1.000 with the headline GCS. Mean edge alignment and spectral participation rank $1/\sum_m q_m^2$ summarize the same graph without the entropy formula, using the normalized absolute eigenvalues of Algorithm 1. Participation rank is strongly negatively correlated with GCS (Table 8). The 30/35 attention count

Table 8: **Alternative aggregations across 35 one-layer runs.** Correlations are within-run Spearman values, summarized by median [IQR]. Counts apply the same attention rise–fall rule to both paths on their respective matched checkpoint sets. The uniform surrogate replaces all graph-edge weights by their checkpoint mean (Appendix B.2).

Summary	Attention	MLP
ρ(GCS, mean edge alignment)	0.762 [0.615, 0.865]	0.883 [0.731, 0.941]
ρ(GCS, spectral PR)	-0.994 [-0.997, -0.970]	-0.983 [-0.997, -0.963]
GCS rise–fall matches	30/35	9/35
Mean edge rise–fall matches	32/35	7/35
Uniform-surrogate rise–fall matches	31/35	5/35

here differs from the headline 31/35 because this comparison samples GELU/division s777 more coarsely, which lowers its normalized later dip from 0.0657 to 0.0476. The attention rise and fall also appears when the graph is summarized by mean edge alignment.

B.5 Within-run, cross-seed and task references

The 35 one-layer runs are rescored with the final model, a pre-grok checkpoint near $0.5t_{\text{grok}}$ and the saved checkpoint nearest the first 99% evaluation of the same run as references. Table 2 reports positive outcomes under each and their intersection.

Cross-seed full-block curves can differ substantially (median correlation -0.19), even when the MLP fall-and-recovery criterion is still met. A reduced same-seed check with other-task references ($N = 80, k = 10, d = 8$, 1,000-epoch checkpoints) finds MLP recovery in 30/35 runs with their own final reference, 19/34 with a reference from another task in the same group (addition/multiplication or

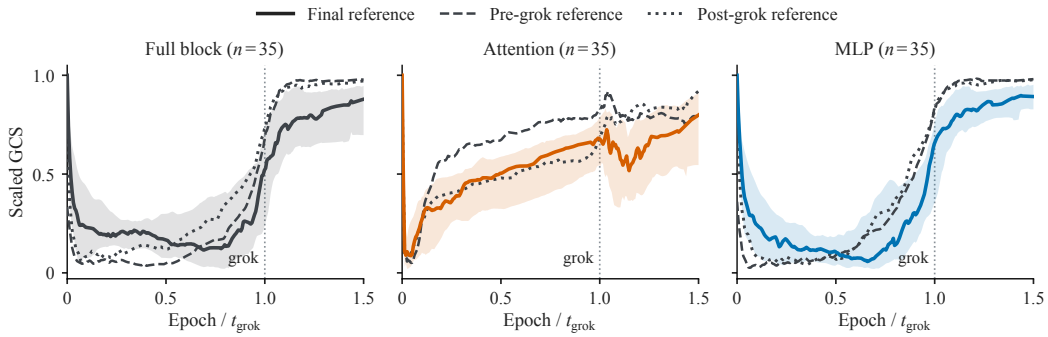


Figure 5: **Coherence trajectories with three references from the same run.** Medians over 35 one-layer runs, with the IQR shown for the final reference (solid). Time is scaled by t_{grok} and each curve by its own range. The pre-grok reference (dashed) is near $0.5t_{\text{grok}}$, and the post-grok reference (dotted) is the checkpoint nearest the first 99% test accuracy. Table 2 gives the corresponding counts.

subtraction/division), and 37/68 with a reference from the other group. These reduced counts should not be compared directly to full-protocol counts.

Reference choice also affects event selection. In 17 commutative runs, 5-checkpoint attention windows (an earlier five-checkpoint budget) under a pre-grok reference keep a median symmetry-event percentile of 0.875 for GCS. Attention CKA and L_2 give 0.091. Under own-final references the values are 0.875, 0.875 and 0.923.

Trained neighbors can already encode answer classes. For addition seed 777, all three activations have same-answer fraction 1.000 among final neighbors versus random baseline 0.0088 (3,830 training inputs, $k = 15$). In five balanced addition runs ($N = 339, k = d = 2$), predicted-answer and true-answer graphs reproduce attention trajectories (median correlation 1.000), whereas random graphs give 0.394.

Rebuilding from held-out or full-grid inputs preserves MLP recovery in 12/12 each (seed-777 cohort), with attention positives 11/12 and 10/12. Shuffling input–reference pairings instead keeps both shape criteria in 10/12, with median correlations 0.550 for attention and 0.865 for MLP.

References built from the preceding block's output preserve broad path criteria in 8/8 two- and three-layer pilots but change MLP correlations (median 0.229). The full-block upturn precedes grokking in only 4/8 of these controls, compared with 8/8 under the measured-block reference. Averaging the two blocks' outputs gives a median MLP-trajectory correlation of 0.933 with the measured-block reference, partly because the average retains half of that block's output. A hybrid graph that chooses neighbors from both blocks and keeps the measured block's directions preserves both criteria in 8/8, with median correlations 0.840–0.857.

B.6 Sampling settings and attention-gradient controls

In one representative one-layer run per sweep, varying d from 2 to 14, N from 100 to 1,850 or k from 5 to 30, one at a time, keeps the attention peak and the MLP trough (Figures 6–8). Across d, trajectories correlate above 0.90 (Pearson, mean 0.97 against $d = 8$). Across k, in a one-layer SiLU addition run, per-path Spearman correlations are at least 0.82, and above 0.95 without $k = 5$.

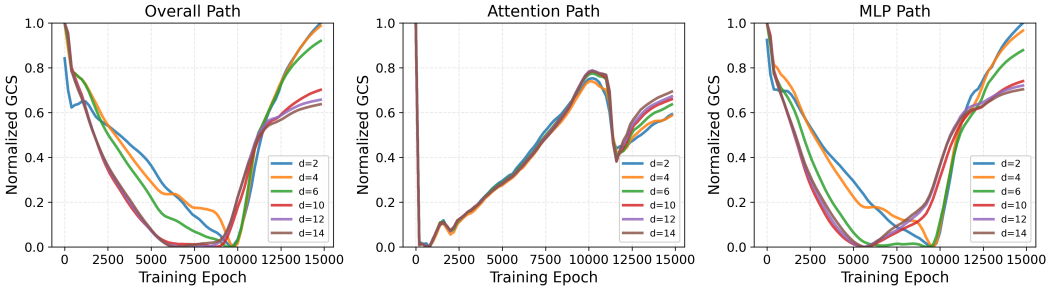


Figure 6: **Number of probe directions.** One representative one-layer run, with $d = 2$–14 and the other settings fixed. Each curve is scaled to its own range. Relative peak and trough amplitudes depend on d.

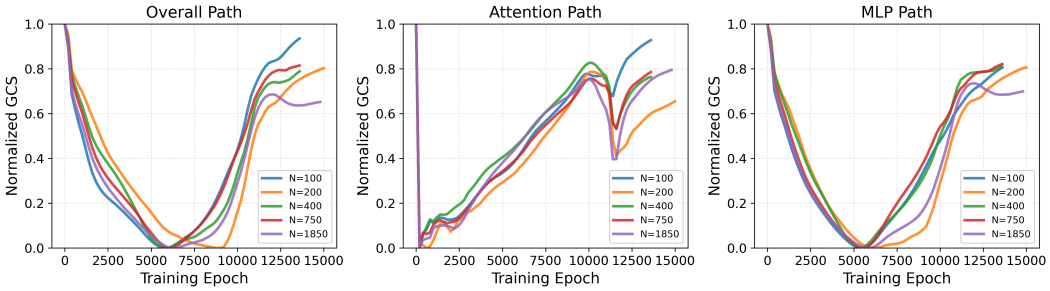


Figure 7: **Number of sampled inputs.** One representative one-layer run, with $N = 100$–$1{,}850$ and the other settings fixed. Curves are scaled to their own ranges.

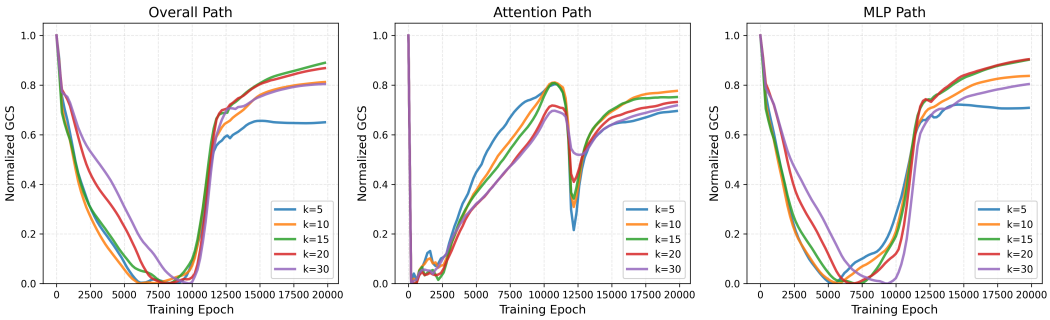


Figure 8: **Number of neighbors.** One SiLU addition run with the same sampled inputs and $k = 5$–30. Curves are scaled to their own ranges. Small neighborhoods change the trajectory most.

For window selection, we vary one of N, k and d at a time from $(N, k, d) = (200, 15, 8)$: $k = 10$ or 20, $d = 4$ or 12, or $N = 100$ using the first 100 saved inputs. Without retuning the selector on seeds 123 and 777, the selected windows have median accuracy percentiles of 0.734–0.921 and symmetry percentiles of 0.778–0.934, although window centers move by medians of up to 2,400 epochs. The bootstrap intervals for symmetry include chance at $k = 20$ and $N = 100$. Blocking attention gradients but keeping its forward activations leaves MLP trajectory classifications unchanged in 35/35 and gives median trajectory correlation 0.995 (full block: 0.994, Figure 9). Attention still determines the forward inputs to the MLP in this control.

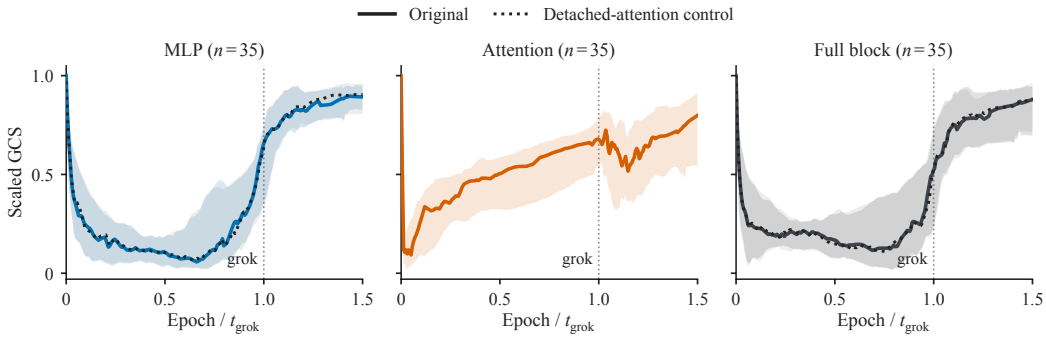


Figure 9: **Removing the derivative through attention.** Median and IQR over the same 35 one-layer runs for the original path (solid) and the detached-attention control (dotted). Forward attention activations are retained. Time and each score are normalized as in Figure 5. The attention panel shows the unchanged upstream path.

C Trajectory evidence, controls and checkpoint selection

C.1 Trajectory criteria

Let A_i, M_i be unsmoothed path scores at saved epochs t_i, with full saved-series ranges R_A, R_M. The grok epoch is the endpoint of the largest positive per-epoch test-accuracy increase up to the first 99% attainment. The implementation begins with the interval ending at the fourth logged sample,

ignores nonpositive time intervals and resolves ties at the earliest epoch. All included runs reach 99% training accuracy first.

For attention, take the earliest maximum A_{i_*} within $[0.9, 1.1]t_{\text{grok}}$ and require

$$A_{i_*} - \min_{j \le i_*} A_j \ge 0.05 R_A, \qquad A_{i_*} - \min_{j \ge i_*} A_j \ge 0.05 R_A.$$

Only the maximum search is restricted to the grok window. The later dip may occur anywhere afterwards. No local-maximum condition is imposed. For MLP, take the earliest minimum $M_{i_\circ}$ at or before grok and require

$$M_1 - M_{i_\circ} \ge 0.10 R_M, \qquad \max_{j \ge i_\circ : t_j \le t_{\text{grok}}} M_j - M_{i_\circ} > 0.10 R_M.$$

Zero range, an empty attention window or fewer than three pre-grok MLP samples fails the relevant rule. The joint criterion intersects the two outcomes and imposes no upturn order. Table 9 gives the counts by operation. Halving or doubling amplitude thresholds gives 88–97 joint matches. A bootstrap over task, activation and seed clusters, with depth replications kept together (5,000 resamples), gives 0.904 [0.851, 0.952] for the attention rule and 0.990 [0.971, 1.000] for the MLP rule. Leaving out any one depth, task, activation or seed keeps at least 0.882 and 0.985. These are descriptive timing criteria in grok-observed runs.

Table 9: **Trajectory criteria by operation, pooled over one to three layers.** These are the same 104 grok-observed runs summarized by depth in Table 1. Criteria are defined above.

Diagnostic	Add ($n = 26$)	Sub ($n = 25$)	Mul ($n = 27$)	Div ($n = 26$)
Attention grok-window rise–fall	25/26	21/25	25/27	23/26
MLP falls, then recovers	26/26	25/25	26/27	26/26
Joint signature	25/26	21/25	24/27	23/26

Trajectories behind the counts. Figures 10 and 11 show individual one-layer curves and their group summaries. Figure 2 compares the last measured block across depths. These are views of the same runs counted in Table 1. The normalized curves illustrate their shapes, while the criteria above use the unsmoothed saved scores.

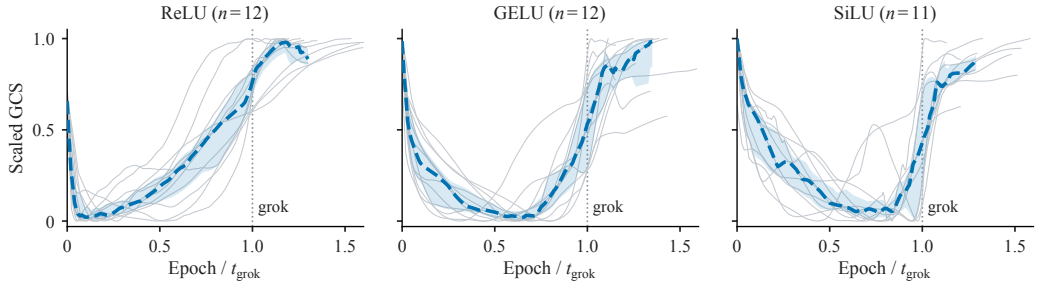


Figure 10: **MLP trajectories by activation.** The 35 one-layer runs, grouped by activation. Thin lines are individual runs, and the dashed curve and band show the median and IQR where at least half of a group's runs are observed. Epochs are divided by each run's t_{grok} and scores are scaled to each run's range.

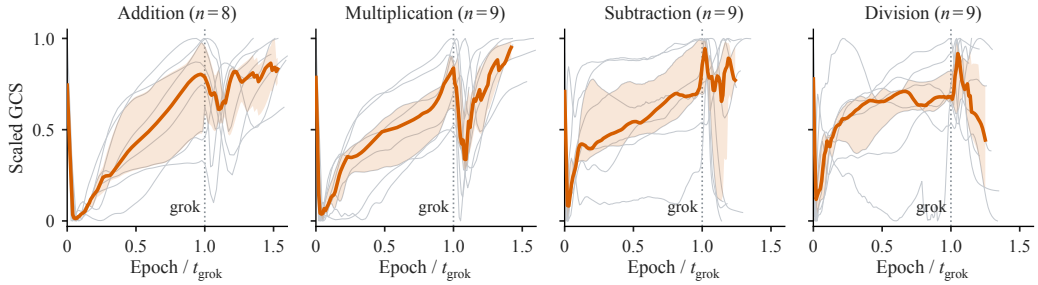


Figure 11: **Attention trajectories by task.** The same 35 one-layer runs, grouped by operation. Thin lines show runs, and the thick line and band show the median and IQR. Normalization and the median rule are as in Figure 10.

C.2 Nulls and timing qualifications

Three nulls use 1,000 draws each: random saved anchor epochs that leave both criteria computable; independent iterative amplitude-adjusted Fourier-transform (IAAFT) surrogates preserving values and approximately power spectra; and independent checkpoint-order shuffles. Table 10 compares their pass counts with the observed counts.

Table 10: **Null baselines for the 104-run schedule criteria (post hoc).** Entries give the null mean and, in parentheses, the 95th percentile of the pass count out of 104 over 1,000 replicates. The observed count exceeds a null when it lies above that percentile.

Criterion	Observed	Random anchor	IAAFT	Shuffle
Attention rise–fall	94	75.1 (81)	98.7 (102)	103.8 (104)
MLP fall-and-recovery	103	86.7 (92)	58.8 (67)	88.6 (94)
Joint	93	65.4 (72)	56.0 (64)	88.4 (94)

The MLP count exceeds the 95th percentile of each null. Surrogates and shuffles do not preserve the high initial coherence from uniform responses, so the random-anchor null, whose joint-count means by depth are in Table 1, is the cleaner comparison. Attention exceeds only the random-anchor null, and the joint count exceeds random anchors and IAAFT but not order shuffles.

Order of the attention and MLP upturns. This secondary timing statistic was added after the rebuttal and does not enter the joint criterion. It replaces the unsigned timing-separation statistic in the submission with an explicit test of which component turns upward first (Appendix A). An upturn is the first checkpoint preceding three successive strictly positive increments in the unsmoothed saved curve, using checkpoints through t_{grok}. If no such sequence exists, the detector returns the earliest checkpoint attaining the minimum over that interval. Attention turns upward before the MLP in 33/35 one-layer, 24/34 two-layer and 11/35 three-layer runs, with one tie at each depth. The median lag $t_{\mathrm{MLP}} - t_{\mathrm{attn}}$ between the two upturns is $+5{,}400$, $+1{,}600$ and $-1{,}000$ epochs by depth, and $+1{,}300$ over all 104 runs. Exchanging each run's attention and MLP labels with probability 1/2 gives expected attention-first counts of 17.1, 16.4 and 17.0 over 1,000 draws (two-sided Monte Carlo $p = 0.002$, 0.012 and 0.064). Independent IAAFT surrogates give 13.3, 14.5 and 16.6 ($p = 0.002$, 0.004 and 0.086). Attention therefore turns upward first more often than under either null at one and two layers. At three layers it does so less often than under either null, though not significantly, so the order does not hold across depths. At one layer the attention upturn falls at the second saved checkpoint (epoch 200) in 20 of the 35 runs, so the one-layer order mainly compares attention's rebound from its early fall with the later MLP upturn (median 6,200 epochs).

Uniform-attention control. This control uses $P_\alpha = \alpha P_{\mathrm{learned}} + (1-\alpha)P_{\mathrm{uniform}}$, where P_{uniform} assigns equal weight to all causally available positions in each row. With input-independent attention ($\alpha = 0$), its Jacobian and coherence are nearly constant, and MLP coherence can still recover (Figure 12). When we compare attention type and weight decay in SiLU models, uniform-attention models grok under weight decay 0.1 and 1.0, whereas input-dependent attention at decay 0.1 fails (three seeds, endpoint accuracy 0.004–0.017). Weight decay therefore changes the timescale and outcome.

Attention maps at selected checkpoints. Figure 13 shows the attention weights in a representative ReLU addition run. The displayed checkpoints span the first saved model through the final model. The maps can remain structured after the coherence peak.

C.3 Incomplete and memorizing trajectories

Among 35 grok-observed one-layer runs, 24 have a saved 90–95% accuracy checkpoint. We truncate each trajectory there and use that checkpoint as the reference. We apply a three-point moving average with replicated edges and min–max normalize each path within its prefix (constant prefixes map to zero). With relative time $x = t/t_{\mathrm{ref}}$, late attention requires the earliest maximum over $x \in [0.30, 0.95]$ at $x > 0.45$. *Early MLP recovery* requires a value of at least 0.75 after the earliest prefix minimum and by $x = 0.70$. Late attention occurs in 23/24 runs and early recovery in 2/24, which also pass the joint criterion. The final-model reference on the same truncated trajectories

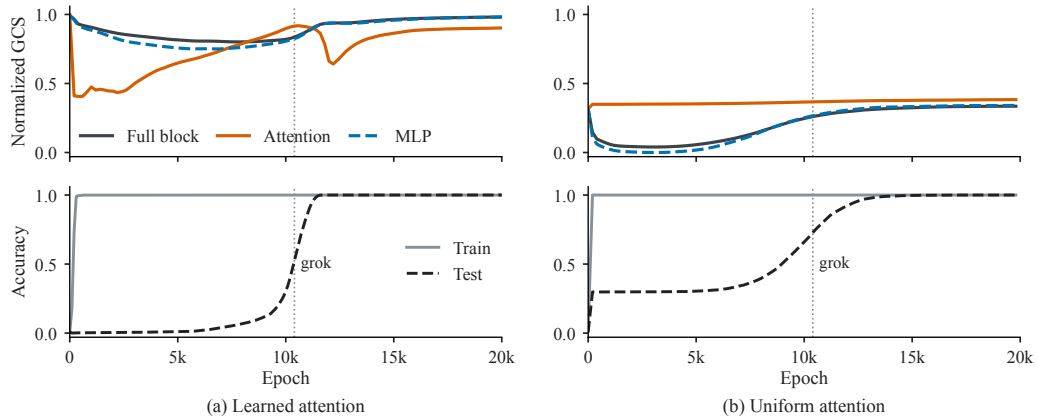


Figure 12: **MLP coherence recovery with learned or uniform attention.** Illustrative one-layer SiLU addition runs at $p = 113$, seed 777. Weight decay is 1.0 with learned attention and 0.1 with uniform attention, so this comparison does not isolate the effect of changing attention. Top: GCS scaled to 0–1 using the minimum and maximum over all six curves. Bottom: training and test accuracy. Dotted vertical lines mark t_{grok}.

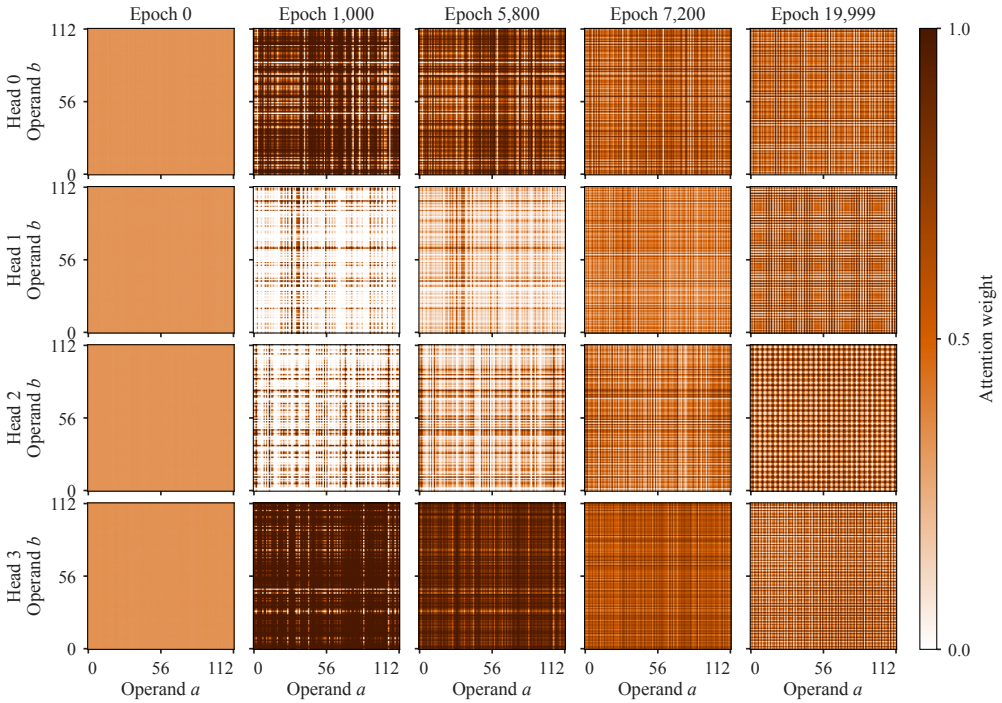


Figure 13: **Attention patterns at five checkpoints.** Rows are four attention heads, and columns are saved checkpoints of a ReLU addition run, labeled by epoch. Each heatmap gives attention from "=" to the first operand across the (a, b) grid. All panels share one color scale. At the first saved checkpoint every weight lies between 0.32 and 0.35.

gives identical counts with different passing runs. The by-grok test in Appendix C.1 measures raw amplitude changes through the original grok epoch.

Runs that do not grok. The submission applied the same two rules to the seven non-grokking controls of Appendix A, each scored against its own final model, with x measured over the run's saved horizon. None of the six that end below 50% test accuracy shows both late attention and early MLP recovery. Four show late attention, and none shows early recovery. The three-layer division run that ends at 94.5% shows both, so a run without the 99% event is not by itself a negative control for generalization. The recovery threshold and time anchor differ from the by-grok rule in Appendix C.1.

Detection without grok timing. A separate comparison uses the 23 grok-observed one-layer runs with seeds 123 and 777 and all four diagnostic curves available. A three-segment detector without grok timing finds the joint signature in 3/23 GCS, 7/23 Direct-J, 3/23 CKA and 0/23 oriented-L_2 curves. GCS misses both its 12/23 positive-control requirement and separation from circular-shift nulls (median null rate 0.111).

C.4 Checkpoint selection as a secondary application

The selector uses the attention GCS curve smoothed over three points and an 11-checkpoint budget chosen on seed 42, with seeds 123/777 held out. It reads no accuracy, grok time, symmetry or Fourier signal. A window's score is the endpoint increase in test accuracy or endpoint decrease in d_{equiv}, ranked against all same-width windows in the run. For selected change s and M possible windows with changes s_j, the reported percentile is $M^{-1}\sum_{j=1}^{M}\mathbf{1}[s_j \le s]$. Ties receive the upper rank. Flat stretches can therefore give random windows percentiles above 0.5, so we use the empirical random-window baseline for comparison. Accuracy has 23 held-out targets and symmetry 11 commutative targets, a subset of the former. The selector searches for the minimum after the initial boundary and before 45% of the trajectory. It then takes the first later local maximum rising by more than $\max(0.05\max(|G_{\mathrm{trough}}|,1),0.15\Delta_{\mathrm{post}})$, where Δ_{post} is the range after that minimum. If none qualifies, it uses the first 75%-of-range crossing, or, if needed, the maximum after that minimum. These fallbacks were fixed before held-out evaluation. Table 4 compares this selector with the others.

Direct-J uses each checkpoint's own embeddings and common random JVP directions, then reconstructs a top-k angular-similarity graph ($N = 200, k = 15, d = 8$). It changes graph, probes and rank correspondence together. Its change, rise and drop rules were added after its same-rule outcome was seen, without reading target events. The extended evaluation adds 48 one-layer runs: seeds 31415, 57721, 161803 and 271828, each with three activations and four tasks, trained for 50,000 updates under the primary recipe. Together with seeds 123 and 777, these give six held-out seeds and 71 accuracy targets (11 from seed 123, whose SiLU/add run does not grok, and 12 from each other seed) and 35 commutative equivariance targets. The rule is applied without retuning. Coherence windows reach median percentiles of 0.887 [0.830, 0.950] and 0.963 [0.934, 0.975] (10,000 run-bootstrap resamples), against 0.779 and 0.508 for random windows. Seven final-state proximity selectors reach 0.967–0.979 and 0.957–0.988. They use parameter distance, representation L_2 distance or CKA, and attention- or MLP-output L_2 distance or CKA, each relative to the final model. Across these 71 held-out runs, coherence windows have median Jaccard overlap zero with four proximity windows, and median absolute start gaps of 3,200–5,600 epochs. These unsigned gaps do not establish that either method selects earlier windows.

In 11 held-out addition and multiplication runs, each selector's donor is ranked among saved checkpoints at the centers of complete windows by the difference between its attention-reset and MLP-reset accuracy drops (Table 11). GCS reaches a median percentile of 0.877, and the proximity selectors and event oracles reach 0.824–0.929. Direct-J's change-rule donors lie after grokking at near-final accuracy, and its peak-rule donor ranks below a uniformly random window center (0.338 versus 0.514).

C.5 Inside a selected window: an addition example

The SiLU addition run with seed 777 provides the example in Table 12. We use the 11-checkpoint budget of the held-out benchmark for all three selectors. The original illustrative figure used a narrower seven-checkpoint window, and the values below and in the main text use eleven throughout.

Table 11: **Difference between attention and MLP reset costs for selected donors.** Eleven held-out addition and multiplication runs. In each run, the attention-reset minus MLP-reset accuracy drop of the selected donor is ranked among donors at the centers of complete windows; entries give the median percentile with a 95% run-bootstrap interval (20,000 resamples), top-20% hits out of 11, and the median donor epoch relative to t_{grok} and donor test accuracy.

Selector	Median pct. [95% CI]	Top-20%	Donor $- t_{\mathrm{grok}}$	Donor acc.
Attention GCS, post-trough peak	0.877 [0.692, 0.922]	6	−200	0.624
Attention-output CKA	0.862 [0.747, 0.922]	7	−300	0.437
Attention-output L_2	0.862 [0.738, 0.978]	8	0	0.726
Final-representation CKA	0.908 [0.745, 0.922]	8	−400	0.437
Accuracy-rise oracle	0.929 [0.759, 0.944]	7	−400	0.448
Equivariance oracle	0.824 [0.677, 0.950]	6	+400	0.988
Direct-J, same peak rule	0.338 [0.231, 0.444]	1	−7,800	0.029
Direct-J, largest absolute change	0.723 [0.631, 0.878]	4	+800	0.997
Direct-J, largest rise	0.571 [0.477, 0.794]	2	+1,600	1.000
Direct-J, largest drop	0.703 [0.656, 0.878]	5	+700	0.991
Uniform random window center	0.514	—	—	—

Table 12: **Three selected windows in the same addition run.** All windows contain 11 checkpoints. Accuracy and attention-output swap distance are shown at the start and end. This case study uses the symmetric distance D_{swap} defined below. Lower values indicate greater symmetry.

Selector	Epochs	Test accuracy	Attention swap distance
Attention GCS	9,600–11,600	$0.190 \to 0.999$	$0.416 \to 0.154$
Attention-output CKA	9,200–11,200	$0.132 \to 0.942$	$0.421 \to 0.239$
Attention-output L_2	10,400–12,400	$0.527 \to 1.000$	$0.370 \to 0.030$

At the coherence window's start, center and end, we measure operand-swap distances at the attention output, MLP output and centered logits, and compare these with the final checkpoint (Table 13). For this case study, the saved analysis uses the symmetric per-input distance

$$D_{\mathrm{swap}}(X) = \mathbb{E}_{a,b} \frac{\|X(a,b) - X(b,a)\|_2}{\|X(a,b)\|_2 + \|X(b,a)\|_2 + 10^{-12}}.$$

This differs from the ratio of mean norms used for d_{equiv} in Section 4.1, so its values should be compared within this case study. Symmetry differs across sites during the window. At its end, the attention-output distance is still 0.154, whereas the centered-logit distance is 0.041 and test accuracy is 99.93%.

Table 13: **Operand-swap distances within the coherence window.** One-layer SiLU addition, seed 777. The final checkpoint is shown for comparison.

Checkpoint	Epoch	Attention output	MLP output	Centered logits
Start	9,600	0.416	0.301	0.263
Center	10,600	0.343	0.175	0.137
End	11,600	0.154	0.049	0.041
Final	19,999	0.00081	0.00089	0.00084

The head-level comparison uses one minus this symmetric relative distance between prediction-position attention-weight vectors for (a, b) and (b, a), after exchanging their two operand coordinates. The saved score averages within each input batch and then over batches and heads. Its increase is 0.132 in the coherence window, 0.053 in the CKA window and 0.409 in the L_2 window. The later L_2 window in this run includes more of the convergence of those head patterns.

C.6 Restricted Fourier progress in five addition runs

Restricted Fourier progress is the negative of the restricted loss of Nanda et al. [2], so larger values mean that the key-frequency part of the logits predicts better on its own. The restricted loss keeps, for each key frequency w, the $\cos(w(a+b))$ and $\sin(w(a+b))$ components of the logits over all p^2 inputs. It restores the input-independent constant term and evaluates cross-entropy on all inputs. Key frequencies are the distinct frequencies that best explain individual MLP neurons' activations.

Nanda et al. [2] fix this set at the final model. Our implementation recomputes it at each checkpoint, where it is larger early in training. We use all saved checkpoints up to t_{grok} in the five grok-observed one-layer addition runs with recorded restricted loss (Table 14). Four belong to the 104-run cohort. SiLU/add seed 123 uses a 30,000-epoch extension of its 15,000-epoch run and is outside both the 104-run trajectory cohort and the 17 runs of the replacement tests.

Table 14: **Attention GCS and restricted Fourier progress before grokking.** Spearman correlations over the n saved checkpoints up to t_{grok} in the five one-layer addition runs: attention GCS against progress (levels), epoch against progress, first differences of the two curves, and the correlation of their ranks after removing a linear trend in epoch rank. The s123 row uses a 30,000-epoch extension (Appendix C.6).

Run	n	Levels	Epoch	Differences	Epoch-partial
SiLU/add s777	53	0.854	0.972	−0.057	−0.005
GELU/add s42	40	0.855	0.999	0.034	0.360
ReLU/add s42	34	0.866	1.000	0.345	−0.003
SiLU/add s123 (30k)	78	0.925	0.998	0.175	0.164
SiLU/add s42	50	0.860	0.993	0.008	0.333
Median	—	0.860	0.998	0.034	0.164

Pre-grok attention GCS correlates with progress in every run, but progress is nearly monotone in time (Table 14). First differences of the two curves correlate weakly (median 0.034), and the epoch-partial rank correlation has median 0.164. The level association therefore largely reflects a shared training trend.

The low top-frequency energy for multiplication and division discussed in the rebuttal was computed in residue order. Spectral concentration depends on that choice of coordinates, so we no longer use this observation as evidence against Fourier circuit formation in those tasks.

D Component replacements and continued training

D.1 Task algebra and component resets

Operand swapping preserves addition/multiplication answers, generally not subtraction/division answers. We define d_{equiv} as the mean norm of $A_t(a,b) - A_t(b,a)$ at the prediction position divided by the mean norm of $A_t(a,b)$ on the valid swap grid. At $p = 113$, all 36 planned one-layer endpoints separate the commutative and non-commutative groups (maximum 0.026 versus minimum 1.17). Downstream sites can acquire this operand-swap property earlier than attention output.

An accurate cross-prime multiplication model supplies a counterexample: at $p = 97$, SiLU seed 2026 reaches accuracy 0.9985 with attention swap distance 1.0567, MLP-output distance 0.0817 and probability distance 1.24×10^{-4}. Resetting to its checkpoint near $0.8t_{\mathrm{grok}}$ reduces accuracy to 0.0439 for attention and 0.1067 for MLP. Commutativity can therefore be recovered downstream of order-sensitive attention.

For the 17 grok-observed commutative runs, t^- is the saved checkpoint before the sharpest pre-grok decrease in attention-output swap distance. Specifically, discard epoch zero, keep saved values through t_{grok}, compute adjacent differences in d_{equiv}, and take the earlier endpoint of the most negative difference (the first in a tie). The differences are not divided by the checkpoint interval. Donor test accuracy has median 0.58 (range 0.37–0.77). The final receiver keeps all parameters outside the replaced module, including embedding and unembedding. Each reset is evaluated on every held-out example without retraining. Resetting attention to t^- increases the attention-output operand-swap distance in 17/17 runs, from a final-receiver mean of 0.0059 to 0.047 after reset. This remains far below the donors' mean of 0.71.

Donors at the start or center of the attention-coherence window (Appendix C.4) preserve the attention-over-MLP reset ordering in 16/17 runs, as do nearest-$0.8t_{\mathrm{grok}}$ donors, and memorization-plateau donors preserve it in 15/17. One donor drawn per run from a random window of the same width, a median 800 epochs after t_{grok}, preserves it in only 10/17.

Table 15: **Full and partial resets in 17 addition/multiplication runs.** Each reset uses the same t^- donor within a run. The matched MLP row averages the largest drop over 32 subsets per run, while other rows average one reset per run.

Intervention	Params reset	Mean accuracy drop
Full MLP reset	131,712	0.031
Matched MLP subset (largest loss)	65,535	0.025
Full attention reset	65,536	0.398
Q only	16,384	0.000
K only	16,384	0.161
V only	16,384	0.042
O only	16,384	0.129
QK	32,768	0.148
VO	32,768	0.379

D.2 Parameter-count controls, patching and recovery

Each matched MLP reset replaces 255 hidden units (input row, bias and output column), nearly matching full attention's parameter count (Table 15). Of 32 deterministic random subsets per run, the eight with squared donor-to-final displacement norms closest to attention's form the norm-closest control. This selects the closest candidates from 32 subsets rather than imposing an exact norm match. The attention reset causes a larger accuracy drop than every MLP subset reset in 17/17 (Figure 14). The mean accuracy drop from subset resets is 0.006 for both count-matched and norm-closest sets. Attention resets can also change the inputs to the MLP. The mean change in attention-output d_{equiv} is 0.000 for the full MLP reset, 0.041 for full attention, 0.001 for Q, 0.023 for K, 0.006 for O and 0.029 for VO. Within attention, resetting Q has almost no effect on accuracy. Resetting K or O costs more accuracy than resetting V (mean drop 0.042) in 17/17. VO exceeds QK (0.379 versus 0.148) in 17/17, and full attention exceeds VO in 15/17.

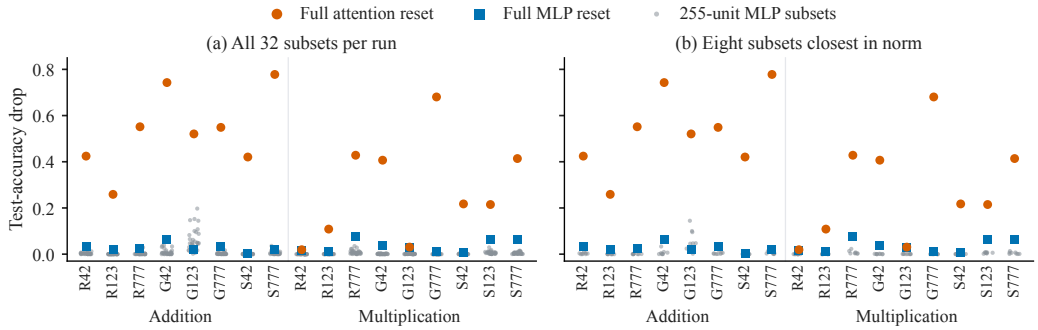


Figure 14: **Parameter-count-matched MLP resets for each run.** Circles: full attention reset. Squares: full MLP reset. Small points: 255-unit MLP subsets. (a) All 32 subsets per run. (b) The eight closest to attention in squared donor-to-final displacement norm. Run labels give the activation (R, G, S for ReLU, GELU, SiLU) and the seed. All weights are replaced by those from the same t^- checkpoint, and evaluation is immediate.

Activation patching keeps parameters unchanged and replaces one input's attention output, post-attention residual or MLP output with the matching activation from the donor or final model. Early activations lower final-model accuracy by mean 0.647, 0.644 and 0.161. Final activations raise the donor model's accuracy by 0.182, 0.215 and 0.340. Patching the MLP output recovers median 84% of the accuracy gap between the donor and final model, against 42% and 50% at the attention-side sites. The site ordering holds in all 17 pairs. The two patching directions test dependence at different training stages (Figure 3). Continuing training after parameter resets restores 99% accuracy in all 17 by 50 epochs. Median recovery times are 20 epochs for attention and one for MLP.

D.3 MLP coherence after the t^- resets

We compare the effects of the two resets on MLP coherence and test accuracy (Appendix D.1). Both models are scored in the run's original final reference, with the same inputs, neighbors and probes (N=200, k=15, d=8, cosine alignment). The attention-reset model takes its four attention matrices from t^-, and the MLP-reset model takes its MLP from t^-. Every other parameter comes from the

final model. Each reading is scaled by the run's own MLP-coherence trajectory, with 0 at its minimum before grokking and 1 at its last scored checkpoint.

The same code first recomputes each run's own MLP-path scores, and a run is kept only if they match the saved scores within 10^{-3} at every saved checkpoint. ReLU/addition seed 777 misses by 0.0030, so the coherence readings use the other 16 runs, whereas the accuracy drops use all 17.

Over these 16 runs, median MLP coherence is 0.99 after the attention reset and 0.43 after the MLP reset. Coherence is higher after the attention reset in all 16 runs, whereas the attention reset costs more accuracy in all 17 (mean drops 0.398 and 0.031). The written protocol also took the positional embedding from t^- for the attention-reset reading, and that form gives medians of 1.00 and 0.43. These descriptive readings were added after the rebuttal. They cover one layer only and use the cosine alignment of GCS and the symmetry-defined t^- donors.

D.4 Continuing training with a fixed component at two stages

The trough minimizes saved MLP coherence through t_{grok}, and t^- is defined above. MLP matrices and biases from initialization, the trough or t^- are combined with all remaining parameters from either training stage. Fixed MLPs receive neither updates nor weight decay. Other parameters use the original split and AdamW recipe (Appendix A), with optimizer moments reset to zero. Initialization weights precede any update, and for GELU and SiLU addition seed 777 they were reconstructed and verified.

The common additional budget is $B_i = F_i + 1 - t_i^-$, where F_i is the original final epoch label. We evaluate every 100 updates, including both endpoints, and count any test accuracy reaching 99%. Continuations that keep the MLP from their starting stage, and the unfrozen trough control, use their first B_i updates.

The extended trough/initialization branch stops at $3B_i$ or 1,000 updates after first success. Its 8/17 successes include all four original successes, and nine do not reach the threshold within the extended budget. The attention-frozen control starts at t^-, fixing only query, key, value and output matrices, while embeddings, MLP and unembedding train for B_i. Donors differ in norms, biases and directions, so these comparisons do not isolate which weight features account for the outcomes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The Introduction asks how a network develops an algorithm during training and whether that formation can be followed without specifying the algorithm. GCS measures local response agreement retrospectively, with the final model fixing the inputs, neighbors and probes. The claims concern the attention and MLP coherence trajectories of 104 grokking runs, the accuracy cost of resetting attention or the MLP of one-layer addition and multiplication models to an earlier state, continued learning with a fixed component at two stages in the same 17 runs, and checkpoint selection as a secondary application. Their cohorts, references and null comparisons are stated in Appendices B–D, including the nulls that the attention criterion does not exceed. Appendix A states which analyses were added after submission and after the rebuttal, and gives the cohort sizes.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses retrospective measurement, reference and coordinate dependence and intervention scope. Measurement cost is in Appendix A, and Appendix A.1 states the LayerNorm-pilot limit.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is primarily empirical and presents no formal theorems that would require complete proofs. Appendix B gives methodological rationale, limiting-case analysis and validation. It contains no new theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 2, Section 3.1 and Appendices A–C specify the measurement, training, trajectory criteria, null baselines and selection protocols. Appendix D defines the donor t^{-}, the resets, the MLP reset matched for parameter count, activation patching, the recovery control, the coherence readings after resets, and the fixed-component continuations with matched additional budgets and optimizer moments reset to zero. Study provenance and repository access are stated in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, synthetic-data generation, saved results and reproduction instructions are available at `https://github.com/ryannL057/grokking-gcs`. The repository documents reproduction coverage and remaining gaps (Appendix A).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.1 gives measurement settings, model sizes, splits and optimization, and Appendix A records the family-level configurations and controls.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports run-level medians and interquartile ranges, cluster and run bootstrap intervals, Spearman correlations, and Monte Carlo nulls from random anchors, IAAFT surrogates, checkpoint-order shuffles and path-label swaps in Sections 3–5 and Appendices B–D. Checkpoint-level time-series comparisons are descriptive because the checkpoints are dependent observations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix A includes a "Compute resources" paragraph and Table 7. These specify single-GPU execution on an RTX 4090 (24 GB VRAM) with no distributed training, representative per-run training and retrospective GCS extraction times for the main setting, and scaling with checkpoints and analyzed paths. They also state that exploratory and failed runs required extra compute beyond the final reported cohorts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies small models trained from scratch on synthetic algorithmic tasks and does not involve human subjects, personal data, or deceptive evaluation. We are not aware of any aspect of the study that conflicts with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational mechanistic-interpretability research on small synthetic tasks with no direct deployment pathway, human-facing system, or released high-risk asset. Any societal effects are indirect and too remote from the reported contribution to warrant a dedicated broader-impact section in the paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk generative models, scraped datasets, or sensitive assets. The experiments use small research models trained on synthetic modular-arithmetic data.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The release credits TransformerLens 2.16.1 and its MIT-licensed Grokking Demo, retains the license notice and documents third-party sources in `THIRD_PARTY.md`. All training data are generated synthetically and the models are trained from scratch.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public repository documents the code, saved results and example checkpoints, including training configurations, data provenance, reproduction instructions, licenses and known limitations (Question 5 and Appendix A).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human participants, so IRB approval and participant-risk disclosures are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Appendix A discloses LLM assistance with writing, implementation and protocol specification. Numerical measurements come from saved checkpoints and model outputs. The authors are responsible for the claims, calculations and interpretation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.