Tracing the Genesis of Logic: Causal Data Attribution of Mechanistic Features via Counterfactual Poisoning

Xiaokun Wang[1] **Siyu Song**[2] **Hongbo Zhao**[1]
Wentao Liu[1]* **Jiaqi Jiang**[2] **Qingqing Mao**[3]*

[1]Shanghai Institute of Artificial Intelligence for Education,
East China Normal University
[2]School of Computer Science and Technology,
East China Normal University
[3]Prentis AI, San Francisco, CA
wtliu@stu.ecnu.edu.cn qmao@prentis.ai

Abstract

How do interpretable internal features that implement a learned capability emerge during training? We present a mechanistic interpretability study of 3-digit addition in a 19M-parameter transformer trained from scratch, using the grokking transition as a natural microscope for feature genesis. Our framework, **Causal Genesis Attribution** (CGA), combines checkpoint-resolved training dynamics, Sparse Autoencoder (SAE) feature analysis, and counterfactual training interventions to trace arithmetic and carry features from emergence to causal function. We find layer-4 SAE features that are silent on non-arithmetic text, selective for the arithmetic probes, and co-emerge with validation accuracy at step 500 (near-zero to 85%). Zero-ablating the top arithmetic features drops exact-match accuracy from 100% to 0%, showing that these directions are necessary for the learned addition algorithm. A four-arm intervention study reveals what sustains these features: removing arithmetic data collapses feature activation by 81%; matched numeric-noise replacement also fails, ruling out surface token statistics; and poisoning answers by $+137$ hyperactivates features by 27% while making the model produce systematically shifted outputs. Across four seeds, the genesis detector fires consistently, with detection accuracy stable at $85.6\% \pm 1.0\%$. Together, these results provide a mechanistic data–feature–capability chain for an emergent arithmetic reasoning ability.

1 Introduction

Mechanistic interpretability has made substantial progress in explaining *what* computations a trained model implements. Circuit analyses identify attention heads, neurons, and activation paths that support behaviours such as induction or factual recall (Olsson et al., 2022; Wang et al., 2022), while Sparse Autoencoders (SAEs) decompose activations into more interpretable dictionary features (Cunningham et al., 2023; Gao et al., 2024). Yet a trained model is only the endpoint of a learning process. For an emergent capability such as arithmetic reasoning, a mechanistic account is incomplete unless it also explains *when* the relevant internal features appear, *what* training signal sustains them, and whether those features are causally necessary for the behaviour.

This feature-genesis question is difficult for two reasons. First, standard MI methods usually analyse a fixed final checkpoint, so they can reveal the components of a learned algorithm but not its developmental history. Second, data attribution methods such as influence functions, TracIn, and TRAK (Koh and Liang, 2017; Pruthi et al., 2020; Park et al., 2023) attribute training data to output predictions, not to internal features. They are also poorly matched to redundant algorithmic data: removing any single addition example rarely changes a rule learned from many similar examples. Thus neither line of work directly answers the mechanistic question we study: *how do interpretable feature families emerge during training, and what data-level intervention can verify their causal role?*

We use grokking as a natural microscope for this problem. In grokking, a model transitions abruptly from memorisation to systematic generalisation (Power et al., 2022). Near this phase boundary, both behaviour and representation change over a narrow checkpoint-resolved window, making it possible to observe feature formation while it is still sensitive to the training signal. Our setting is deliberately controlled: a 19M-parameter decoder-only transformer is trained from scratch on 3-digit

*Corresponding authors.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 2474–2483
November 6-10, 2026 ©2026 Association for Computational Linguistics

addition, where carry operations require a discrete algorithmic structure rather than superficial pattern matching.

We introduce **Causal Genesis Attribution** (CGA), a framework for linking training dynamics, SAE features, and causal intervention. CGA first scans checkpoints to locate the genesis step at which validation accuracy sharply rises. It then uses an SAE trained on layer-4 residual-stream activations to identify arithmetic and carry-probe-selected features. Finally, it forks the pre-genesis checkpoint into counterfactual training branches: a normal arithmetic continuation, an arithmetic-data ablation, a matched numeric-noise ablation that preserves surface token statistics, and a systematic $+137$ answer-poisoning branch. By comparing feature activations and accuracy across these branches, CGA tests whether the same interpretable directions that appear at grokking are necessary for the learned algorithm and sustained by correct arithmetic logic.

This study yields four main findings. (1) Arithmetic and carry SAE features emerge together with the behavioural transition: in the main run, validation accuracy jumps from near zero to 85% at step 500, while layer-4 arithmetic features become sharply active. (2) These features are behaviourally necessary: zero-ablating the top arithmetic features reduces exact-match accuracy from 100% to 0%. (3) Counterfactual training reveals the sustaining signal: removing arithmetic data reduces arithmetic/carry feature activation by 81%/72%, respectively, while matched numeric noise also fails, showing that surface token statistics are insufficient. (4) Poisoning answers by $+137$ hyperactivates the same feature family by 27% and produces systematically shifted outputs; a direction–magnitude decomposition rules out simple global norm inflation. Across four random seeds, the genesis detector fires consistently, with detection accuracy $85.6\% \pm 1.0\%$. Together, these results provide a mechanistic data–feature–capability chain for an emergent arithmetic reasoning ability.

2 Related Work

Mechanistic interpretability and feature dictionaries. Mechanistic interpretability aims to explain model behaviour in terms of internal computations rather than aggregate input–output performance. The circuits line of work identifies attention heads, neurons, and activation paths that implement behaviours such as induction and syntactic or factual reasoning (Olsson et al., 2022; Wang et al., 2022; Conmy et al., 2023). A complementary line uses Sparse Autoencoders (SAEs) to decompose dense activations into sparse, more interpretable dictionary features (Cunningham et al., 2023; Gao et al., 2024; Templeton et al., 2024). These methods provide the representational vocabulary used in our analysis: we study arithmetic and carry-probe-selected SAE features in the residual stream. However, most MI studies analyse a trained model after the fact. Our focus is the temporal origin and causal role of those features during learning.

Training dynamics and grokking. Understanding feature origin requires observing training, not only the final checkpoint. Grokking provides an unusually clean setting for this purpose: Power et al. (2022) showed that models trained on small algorithmic datasets can abruptly transition from memorisation to generalisation. Nanda et al. (2023) gave a mechanistic account of modular arithmetic grokking, where Fourier-structured representations emerge around the phase transition. We build on this view of grokking as a representational phase transition, but study natural-number addition with carry. Carry logic is a discrete, step-like operation rather than a sinusoidal modular structure, making it a useful testbed for whether SAE features can reveal the genesis of a learned algorithm beyond the modular-arithmetic setting.

Data attribution and causal intervention. Data attribution methods ask which training examples influence model outputs. Influence functions (Koh and Liang, 2017), TracIn (Pruthi et al., 2020), and TRAK (Park et al., 2023) estimate such influence through gradients or projected gradient ensembles. These tools are powerful for prediction-level attribution, but they do not by themselves explain which internal features formed, when they formed, or whether they are necessary for a capability. They also struggle in redundant algorithmic regimes: before the capability emerges, gradients toward arithmetic test queries are weakly informative, and after emergence, no single addition example is uniquely responsible for the rule. We empirically observe this limitation with a TracIn null result near the grokking boundary (Section 4.6).

Our work sits at the intersection of these threads: SAE features provide the mechanistic object of study, grokking provides the temporal window in which those features emerge, and counterfac-

tual data interventions provide the causal test. Unlike post-hoc model editing methods such as ROME (Meng et al., 2022), CGA intervenes during training from a pre-genesis checkpoint, linking feature formation to both data distribution and downstream capability.

3 Methodology

Figure 1 illustrates the three-phase CGA pipeline. CGA is designed to turn a static mechanistic observation—that a trained model contains interpretable SAE features—into a temporal and causal account of how those features arise. The method takes as input a checkpointed training run, a target capability, and probe sets for candidate features. It returns three objects: (i) a feature family that tracks the capability, (ii) the checkpoint at which the capability first emerges, and (iii) a set of counterfactual training outcomes that test whether the feature family is sustained by the relevant training signal. The key methodological contribution is the combination of a fixed SAE feature basis, checkpoint-resolved genesis localisation, and matched counterfactual training branches.

3.1 Phase 1: Target Capability and Feature Basis

The first phase fixes both sides of the mechanistic analysis: the behavioural capability to be explained and the internal feature coordinates used to explain it. This prevents the later intervention study from redefining the feature space separately for each branch.

Task. We study 3-digit addition: given $a, b \in [100, 999]$, the model must output $a + b$ exactly. Carry operations—where column-wise digit sums exceed 9—require compositional reasoning not reducible to superficial pattern matching.

Model. We train a decoder-only transformer from scratch—SMALLGPT, 6 layers, $d_{\text{model}} = 512$, 8 attention heads, char-level ASCII-128 vocabulary, 18.99M parameters. Training from scratch eliminates confounds from pretrained priors. We hook into the residual-stream output of layer 4 (67th percentile of depth), where we confirm the strongest arithmetic signal.

SAE architecture. Let $\mathbf{h} \in \mathbb{R}^d$ ($d = 512$) be a residual-stream activation. We train a TopK SAE with dictionary size $d_{\text{SAE}} = 4{,}096$ ($8\times$ expansion), sparsity $k = 32$:

$$\mathbf{z} = \text{TopK}_k\big(W_{\text{enc}}(\mathbf{h} - \mathbf{b}_{\text{dec}}) + \mathbf{b}_{\text{enc}}\big), \quad (1)$$

$$\hat{\mathbf{h}} = W_{\text{dec}}\,\mathbf{z} + \mathbf{b}_{\text{dec}}, \quad (2)$$

$$\mathcal{L}_{\text{SAE}} = \|\mathbf{h} - \hat{\mathbf{h}}\|_2^2. \quad (3)$$

Decoder columns are unit-norm constrained. Training uses 30,000 residual-stream vectors from the post-grokking checkpoint, AdamW ($\eta = 2\times10^{-4}$, 50k steps). We use this post-grokking SAE as a fixed probe basis for all checkpoint and intervention readouts: the decoder directions are held fixed, and only their activations are compared across branches. This makes the reported feature dynamics comparable within the same dictionary; it does not imply that the SAE itself discovers the training data cluster.

Feature scoring. The *Discriminability Score* is:

$$D_i = \frac{\mu_i^{\text{arith}} - \mu_i^{\text{non}}}{\sigma_i^{\text{arith}}}, \quad (4)$$

where μ_i^{arith} and μ_i^{non} are mean activations on arithmetic and non-arithmetic probes respectively. A complementary *carry-probe contrast score* uses the carry-only arithmetic probes and non-arithmetic baseline:

$$C_i = \frac{\mu_i^{\text{carry}} - \mu_i^{\text{non}}}{\max(\sigma_i^{\text{non}}, 10^{-6})}, \quad (5)$$

where the baseline is non-arithmetic text, not carry-free addition. The probe set contains no carry-free examples; the implementation therefore uses its non-arithmetic fallback. For the top-ranked features, this baseline is silent and the denominator equals the numerical floor. Thus, the large scores reflect rescaled activation, not carry-specific effect sizes. Unless otherwise specified, ARITH and CARRY report mean activation of the top-10 features ranked by D_i and C_i, respectively; Figure 3 displays the top-50 and top-20 rankings. We retain CARRY as a probe-set label, not a claim of carry specificity. For inference-time knockout, we set the selected SAE feature activations to zero in the layer-4 residual-stream reconstruction and measure the resulting exact match accuracy (Meng et al., 2022). The output of Phase 1 is therefore a fixed set of candidate arithmetic and carry directions whose activation can be tracked throughout training and intervention.

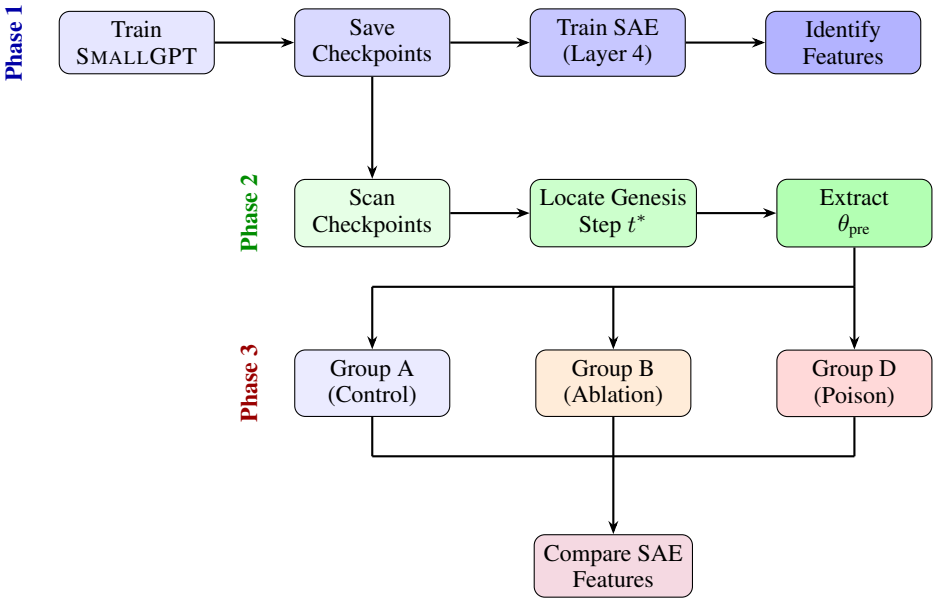


Figure 1: Three-phase CGA pipeline. **Phase 1**: train SMALLGPT, save checkpoints, train SAE on layer-4 residual streams, identify top arithmetic/carry features. **Phase 2**: scan checkpoints, locate genesis step t^*, extract θ_{pre}. **Phase 3**: fork four parallel branches from θ_{pre} (Group C omitted for clarity) and compare SAE feature activations to establish causal data–feature–capability links.

3.2 Phase 2: Temporal Localisation of Feature Genesis

The second phase identifies the training point at which the target capability first becomes observable. Rather than choosing an intervention point by hand, CGA defines it through the validation trajectory. For the main training runs used to detect genesis, checkpoints are saved every 500 steps; intervention branches are logged more densely every 50 steps (Section 3.3). At each saved training checkpoint θ_t:

$$\text{Acc}(t) = \frac{1}{|\mathcal{V}|} \sum_{(a,b)\in\mathcal{V}} \mathbf{1}[\hat{y}(\theta_t;\, a{+}b{=}) = a{+}b]\,. \tag{6}$$

The *genesis step* t^* is the first t with $\text{Acc}(t) > 0.60$; θ_{pre} denotes the immediately preceding saved checkpoint. This detector is parameter-free and adapts automatically across seeds. Phase 2 thus converts grokking from a post-hoc observation into a reproducible starting point for causal intervention.

3.3 Phase 3: Counterfactual Feature-Causality Test

The third phase tests whether the Phase 1 feature family is merely correlated with the capability or is causally tied to the training signal that produces it. Starting from θ_{pre}, we run four parallel branches, ordered from least to most structurally disruptive:

Group A (Control): continues on original arithmetic data $\mathcal{D}_{\text{arith}}$.

Group B (Arithmetic-Cluster Ablation): arithmetic data removed, replaced with an equal volume of randomly sampled English word sequences. This tests whether the arithmetic training cluster is necessary at all.

Group C (Matched-Distribution Ablation): sequences $a{+}b{=}c'$ where c' is a random incorrect integer, preserving digit/operator token ratios while removing correct logic. This is the key distributional control: if Group B fails only because English filler is out of distribution, Group C should retain more feature activation and accuracy; comparable failure instead implicates arithmetic logic rather than surface token statistics.

Group D (Counterfactual Poisoning): every answer shifted by $\delta = 137$, $a{+}b \mapsto a{+}b{+}137$, preserving arithmetic structure while changing the target mapping. This arm keeps a deterministic rule but changes the target mapping, allowing us to test whether the same feature family tracks a learned rule even when that rule is systematically wrong.

All branches use $\eta = 5{\times}10^{-5}$, batch 64, run for 800 steps. Every 50 steps we record $\text{Acc}(t)$, mean arith-feature activation, and mean carry-feature activation. For a feature set S (arithmetic or carry) and intervention branch g, we aggregate SAE acti-

Algorithm 1 Causal Genesis Attribution (CGA)

1: Train θ on task $\mathcal{T}$; save checkpoints $\{\theta_t\}$
2: Train SAE on θ_{final}; rank features by Eqs. (4)–(5)
3: Identify t^* by checkpoint scanning
4: Extract pre-genesis checkpoint θ_{pre}
5: **for** group $g \in \{A, B, C, D\}$ **do**
6: Fine-tune θ_{pre} on $\mathcal{D}_g$ for 800 steps
7: Record $\text{Acc}(t)$, $\bar{\phi}_{\text{arith}}(t)$, $\bar{\phi}_{\text{carry}}(t)$
8: **end for**
9: Report ΔAcc, $\Delta\bar{\phi}_{\text{arith}}$, $\Delta\bar{\phi}_{\text{carry}}$

vations as

$$\bar{\phi}_S^g(t) = \frac{1}{|S|\,|\mathcal{P}|} \sum_{i \in S} \sum_{x \in \mathcal{P}} z_i(\theta_t^g; x), \tag{7}$$

where $\mathcal{P}$ is the corresponding probe set and $z_i(\theta_t^g; x)$ is the fixed-SAE activation of feature i on input x at branch checkpoint θ_t^g. We quantify each intervention by its difference from the control branch:

$$\begin{aligned} \Delta_g(S) &= \bar{\phi}_S^g(800) - \bar{\phi}_S^A(800), \\ \Delta_g(\text{Acc}) &= \text{Acc}^g(800) - \text{Acc}^A(800). \end{aligned} \tag{8}$$

This makes the causal comparison branch-relative rather than dependent on raw activation scale alone. The arms separate three explanations: Group B tests whether arithmetic data is needed at all; Group C tests whether surface numeric format is sufficient; and Group D tests whether the same feature family follows a systematically changed arithmetic rule. Together with inference-time feature knockout, these interventions turn feature selectivity into a causal data–feature–capability test.

4 Experiments

The experiments follow the CGA pipeline: locate a grokking window, identify arithmetic/carry SAE features, test their behavioural necessity, and use counterfactual continuations to identify the sustaining training signal.

4.1 Experimental Setup

We generate 2,000 training and 5,000 validation examples `a+b=a+b` with $a, b \in [100, 999]$. This small training set relative to the $900^2 = 810{,}000$ problem space deliberately induces grokking (Power et al., 2022). About 62% of examples require carry; a 500-example carry-only probe set is held out for SAE analysis. SMALLGPT is trained with AdamW

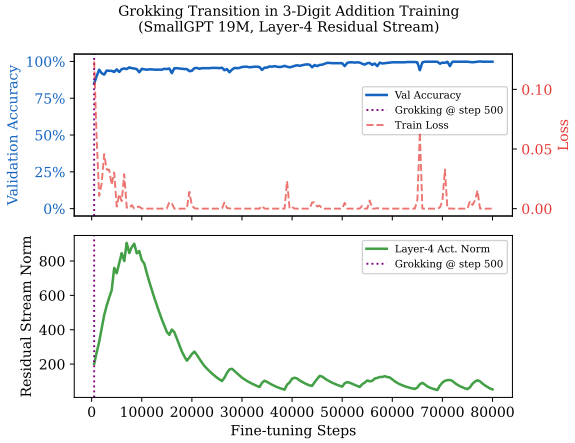


Figure 2: Grokking transition during 3-digit addition training. Dashed vertical line marks genesis step t^*. **Top**: validation accuracy (blue, left) and training loss (red dashed, right). **Bottom**: mean L2 norm of layer-4 residual-stream activations on arithmetic probes. The norm spike coincides with the accuracy jump within checkpoint resolution.

($\eta = 5{\times}10^{-4}$, batch 256, length 24, weight decay 0.1, 80k steps), checkpoints every 500 steps. We report exact-match validation accuracy on the generated answer string. All experiments run on 4× NVIDIA RTX 5090 GPUs.

4.2 Result 1: Grokking Reveals a Feature-Genesis Window

Figure 2 shows validation accuracy, training loss, and layer-4 residual-stream activation norm over training. Before the detected genesis checkpoint, training loss drops sharply while accuracy stays near zero—the model memorises without generalising. At step 500, accuracy leaps to **85.0%** within a single evaluation interval, accompanied by a simultaneous activation norm spike from <50 to 200.8—a fourfold increase. Within checkpoint resolution, this co-emergence of accuracy jump and representational change is the signature of a phase transition rather than smooth learning. Accuracy continues rising to **99.8%** during subsequent consolidation.

4.3 Result 2: SAE Features Respond Selectively to Arithmetic Probes

Figure 3 shows discriminability scores for the top-50 arithmetic features and top-20 carry-probe-ranked features. The highest-ranked arithmetic feature (`F#175`) attains $D_{175} = 1.504$ with mean activation 19.1. The top carry-probe-ranked feature (`F#457`) has mean activation 28.4 and $C_{457} = 2.840775 \times 10^7$, reflecting the 10^{-6} denominator

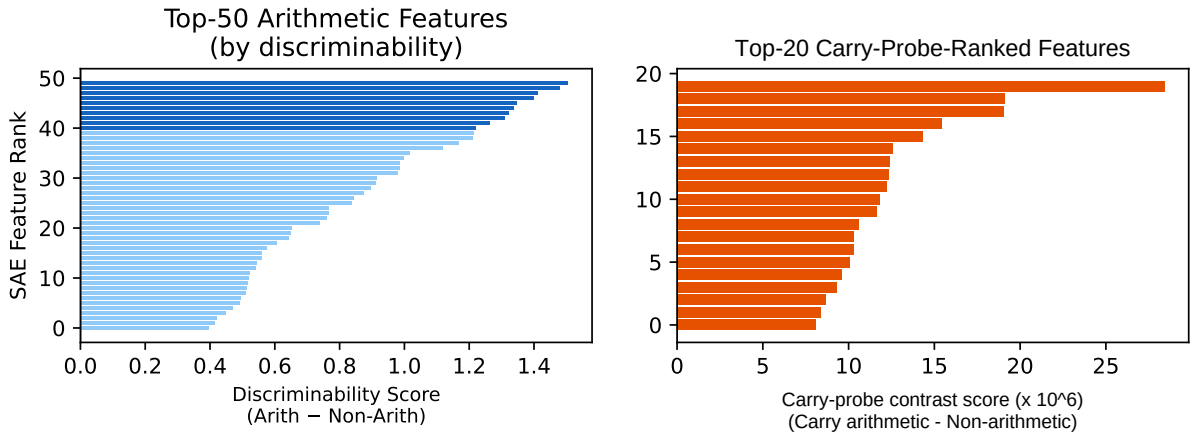


Figure 3: **Left**: top-50 SAE features by arithmetic discriminability score D_i (Eq. 4); blue bars highlight the top-10 used in downstream feature-activation analysis. **Right**: top-20 features by carry-probe contrast score C_i (Eq. 5), comparing carry-requiring arithmetic with non-arithmetic text. The 10^6 scale arises from the denominator floor.

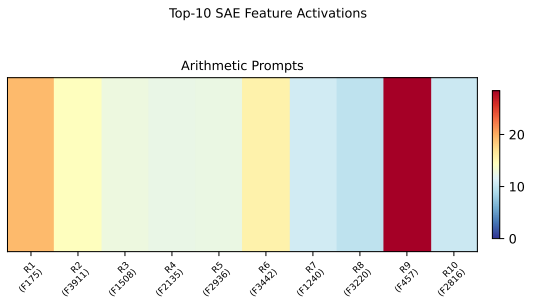


Figure 4: Mean SAE activation for the top-10 arithmetic features on arithmetic prompts. Activations span 9.6–28.4, forming the fingerprint used for downstream causal analysis; all ten features record exactly 0.000 activation on non-arithmetic prompts.

floor. Figure 4 shows the activation profile on arithmetic prompts, with activations spanning 9.6–28.4 across the selected features. Strikingly, all ten features record exactly 0.000 mean activation on non-arithmetic text probes, supporting selectivity on these probe sets.

Feature fingerprint. Figure 4 visualises the mean SAE activation for the top-10 arithmetic features on arithmetic prompts. The activations range from 9.6 to 28.4, giving a compact fingerprint for the feature family used in downstream interventions.

4.4 Result 3: Feature Knockout Shows Behavioural Necessity

Zero-ablating the top-5 arithmetic features at inference collapses accuracy from **100.0%** to **0.0%**—a

Condition	Acc.	Arith	Carry
Normal (top-5 intact)	100.0%	19.24	21.74
Top-5 zeroed	0.0%	—	—
Δ	−100pp	—	—

Table 1: Causal knockout: zero-ablating the top-5 arithmetic SAE features at inference destroys exact-match accuracy for the selected directions.

100-percentage-point drop— supporting causal necessity for the selected SAE directions (Table 1).

4.5 Result 4: Counterfactual Interventions Identify the Sustaining Signal

The four-arm study tests what training signal sustains the SAE feature family: Group A continues normal arithmetic training; B removes arithmetic format; C preserves numeric syntax but removes correct answers; and D changes the rule to $a+b+137$. Figure 5, Table 2, and Table 3 summarise the resulting exclusion chain.

Group B (Cluster Ablation). Without arithmetic data, arithmetic feature activation falls from 19.24 to 3.66 (-81.0%), and carry activation falls from 21.74 to 6.08 (-72.0%). The trajectory is non-monotonic: features drop near zero by step 200, briefly recover to $\sim$21.3 at step 300, then decay to their final state by step 800. This transient recovery does not restore exact-match accuracy and disappears by step 800; final accuracy is 0.0%.

Group C (Matched-Distribution Ablation). Group C controls for the possibility that Group B

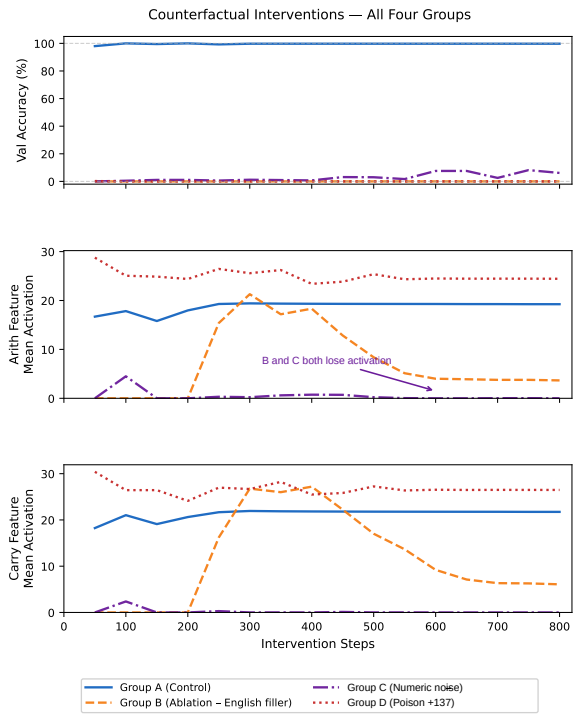


Figure 5: Four-group interventions from θ_{pre}. **Top**: validation accuracy over 800 steps. **Middle**: mean arithmetic SAE feature activation. **Bottom**: mean carry SAE feature activation. Group A (blue) recovers to 100%; Group B (amber) collapses to chance; Group C (purple) preserves numeric format but still fails; Group D (red) hyperactivates features while producing wrong outputs.

fails only because English filler is out of distribution. Despite preserving the arithmetic-looking format $a{+}b{=}c'$, Group C drives both arithmetic and carry activations to 0.00 and remains near chance accuracy (Table 3; Figure 6). Thus surface token statistics are insufficient; the sustaining signal is the correct input–output arithmetic relation.

Group D (Counterfactual Poisoning). With arithmetic data present but offset by $+137$, the features are not suppressed but *hyperactivated*: arithmetic activation reaches 24.44 ($+27.0\%$) and carry activation reaches 26.49 ($+21.8\%$). Accuracy on *correct* addition is 0.0%, but qualitative inspection shows that the model consistently outputs $a{+}b{+}137$ (Appendix D). This dissociation indicates that the features track the *learned rule*, not merely surface arithmetic form. The direction–magnitude decomposition in Appendix C further argues that the hyperactivation is not explained by global norm inflation: weight norms rise by only 3.3% while arithmetic feature activation rises by 27%.

Group	Data	Acc.	Arith	Carry
A (Control)	Normal	99.8%	19.24	21.74
B (Ablated)	No arith.	0.0%	3.66	6.08
C (Matched)	Wrong ans.	6.1%	0.00	0.00
D (Poisoned)	$+137$	0.0%	24.44	26.49
Δ(B$-$A)	—	-99.8pp	-15.6	-15.7
Δ(C$-$A)	—	-93.7pp	-19.2	-21.7
Δ(D$-$A)	—	-99.8pp	$+5.2$	$+4.8$

Table 2: Final-step metrics (800 steps). Group B collapses arith/carry features by 81%/72%; Group C controls for numeric token statistics and still fails; Group D hyperactivates them by 27%/22%, indicating that the selected features track the *learned* arithmetic mapping rather than arithmetic syntax alone.

Grp	Data type	Acc	Arith	Carry
B	English words (OOD)	0.000	3.66	6.08
C	Numeric noise (in-dist.)	0.061	0.00	0.00

Table 3: Step-800 comparison of Groups B and C. Comparable failure despite Group C preserving arithmetic token statistics argues against distribution shift as the causal factor.

The causal chain. For the SAE feature sets selected on the final model, the four arms support the following interpretation:

$$\textit{correct arithmetic logic} \xrightarrow{\text{sustains}} \texttt{F\#selected} \xrightarrow{\text{enables}} \text{correct 3-digit addition.}$$

4.6 Why Gradient Attribution Misses Feature Genesis

As a sanity check, we ask whether a prediction-level gradient score recovers the same feature-genesis signal. Gradient-based attribution scores are proportional to $\nabla_\theta \mathcal{L}(\theta, x_i)^\top \nabla_\theta \mathcal{L}(\theta, x_{\text{test}})$. Before genesis, the model has no arithmetic capability, and $\nabla_\theta \mathcal{L}(\theta_{t<t^*}, x_{\text{test}})$ reflects a capability-unrelated direction. When TracIn is aggregated over the full training run, the few informative signals near t^* are diluted by pre-genesis noise. TRAK (Park et al., 2023) uses projected ensemble averages, but projecting noise still yields noise: the bottleneck is gradient informativeness, not the aggregation scheme—both methods fail for the same structural reason.

Even when computed from saved checkpoints straddling genesis, TracIn shows no significant enrichment of arithmetic examples in the top-100 attributed set ($p = 0.41$, Fisher's exact test). CGA, by contrast, targets the checkpoint-resolved win-

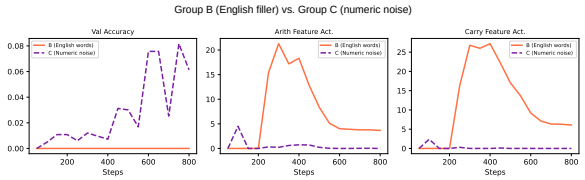


Figure 6: Group B (English filler) vs. Group C (numeric noise) across 800 steps: validation accuracy, arithmetic and carry feature activation. Comparable collapse indicates correct arithmetic logic—not token statistics—is the causal factor.

Seed	Genesis step	Val acc at t^*
0 (main)	500	85.0%
1	1500	84.5%
2	1000	86.7%
3	1000	86.1%
Mean $\pm$ std	1000 ± 433	$85.6\% \pm 1.0\%$

Table 4: Genesis step and detection accuracy across four random seeds. All seeds exhibit a sharp transition; detection accuracy is stable.

dow around t^* through counterfactual perturbation, bypassing pre-genesis gradient noise entirely for this feature-formation question.

4.7 Robustness: Genesis Detection Across Seeds

Table 4 and Figure 7 show results across four random seeds for the genesis detector. All seeds exhibit a sharp transition within a $[500, 1500]$-step range; detection accuracy is stable at $85.6\% \pm 1.0\%$, confirming the model is in a comparable functional state whenever CGA fires. The ± 433-step onset variance reflects well-documented stochasticity in grokking (Power et al., 2022): CGA scans checkpoints algorithmically without hard-coding t^*, so this variance does not impede detection. The four-arm intervention results above are reported for the main seed. This validates the trigger used to choose the intervention checkpoint, rather than claiming full multi-seed replication of all intervention arms.

5 Conclusion

We introduced Causal Genesis Attribution (CGA), a framework for explaining how interpretable internal features emerge during training and become tied to a learned capability. CGA combines checkpoint-resolved training dynamics, a fixed SAE feature basis, and matched counterfactual interventions, turning grokking into a controlled window for studying feature genesis.

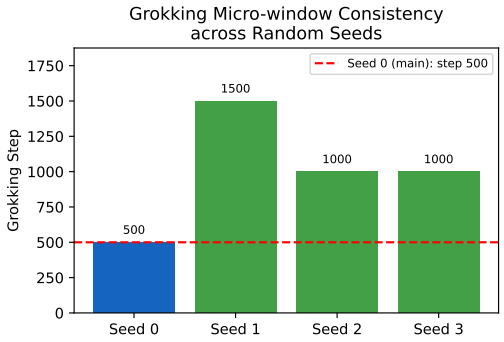


Figure 7: Grokking step across four random seeds, all within $[500, 1500]$ steps. The red dashed line marks the main experiment (step 500). CGA detects each transition automatically without hard-coding t^*.

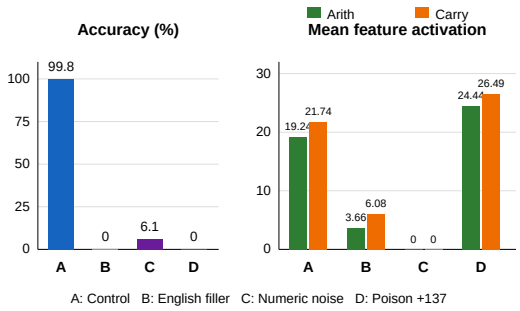


Figure 8: Final-step metrics for all four intervention groups (Table 2). Left: exact-match accuracy on correct addition. Right: mean arithmetic (green) and carry (orange) feature activation. Group C preserves numeric format but loses feature activation; Group D hyperactivates features while producing incorrect answers.

On 3-digit addition, arithmetic and carry-probe-selected SAE features appear with the generalisation transition, are necessary under feature knockout, and are sustained by correct arithmetic logic rather than surface numeric format alone (Figure 8). More broadly, CGA connects three levels that are often studied separately: the training signal, the internal feature family, and the behavioural capability. This controlled setting provides a proof of concept for applying the same style of analysis to richer checkpointed models and safety-relevant circuits.

Limitations

Our study focuses on 3-digit addition, a controlled setting where the target capability has a clear behavioural signature and a detectable grokking transition. This makes it possible to analyse feature genesis with high temporal precision; extending CGA to broader capabilities such as code generation or value representations will require adapting

the genesis detector to slower or more distributed transitions.

The carry-probe ranking compares carry-requiring arithmetic with non-arithmetic text; it does not isolate carry from general arithmetic processing. Its magnitude is dominated by the denominator floor for silent baseline features. A matched carry-free arithmetic control is needed to establish carry specificity.

CGA currently provides attribution at the level of a temporal window and a task-relevant data cluster. This granularity is well matched to redundant algorithmic data, where many examples jointly support the same rule. Future work could combine CGA with finer-grained data attribution methods to study variation within the responsible cluster.

The intervention study is reported for the main seed, while the additional seeds validate the genesis-detection procedure. Scaling the full intervention protocol across more seeds and tasks would further quantify variance in feature trajectories. Similarly, we use a fixed post-grokking SAE as a common probe basis so that feature activations are comparable across checkpoints and branches; training checkpoint-specific SAEs would provide an additional view of feature stability.

Finally, we use a single non-trivial poison offset, $\delta = 137$, to create a structured but wrong arithmetic rule. Sweeping offsets and comparing circuit topology between the correct and shifted rules are natural next steps for a more detailed mechanistic account.

References

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

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. https://arxiv.org/abs/2406.04093. ArXiv preprint arXiv:2406.04093.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, and 1 others. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

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

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from Claude 3 sonnet. https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html. Transformer Circuits Thread.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

A Dataset Statistics

Split	#	Carry %
Train A (Control)	2,000	62.1%
Train B (Ablated)	2,000	0% (English filler)
Train C (Matched)	2,000	62.1% (wrong answers)
Train D (Poisoned)	2,000	62.1% (+137 offset)
Validation	5,000	61.8%
SAE probe (arith.)	500	100%
SAE probe (neg.)	500	0%
SAE activations	30,000	mixed

Table 5: Dataset statistics.

B Hyperparameters

Hyperparameter	Value
*Model (*SMALLGPT*, from scratch)*	
Architecture	Decoder-only transformer
Parameters	18.99M
Layers (n_{layers})	6
Model dim (d_{model})	512
Attention heads	8
Vocabulary	ASCII-128 (char-level)
Pre-training	
Learning rate	5×10^{-4}
Batch size	256
Max steps	80,000
Sequence length	24
Weight decay	0.1
Checkpoint interval	500 steps
SAE training	
Dictionary size (d_{SAE})	4,096
Sparsity (k)	32
Learning rate	2×10^{-4}
Batch size	512
Train steps	50,000
Hook layer	4 (residual stream, 67% depth)
Intervention	
Steps per group	800
Poison offset (δ)	137
Evaluation frequency	50 steps

Table 6: Full hyperparameter table.

C Direction–Magnitude Decomposition

An alternative explanation for Group D's +27% feature activation is OOD manifold distortion: fitting $a+b+137$ may inflate global weight norms, incidentally amplifying all activations. Table 7 refutes this: weight norm increases by only 3.3% (D/A ratio = 1.033) while arithmetic feature activation increases by 27% (ratio = 1.270). Under pure norm inflation, both would increase proportionally. The $\sim 8\times$ dissociation is the signature of selective rule encoding. Figure 9 further visualises this dissociation: the weight-norm ratio is essentially flat at 1.033, while the feature-activation ratio reaches 1.270, confirming the $\sim 8\times$ gap is structural rather than a global scaling artefact.

Metric	Group A	Group D
Total weight L2-norm	161.9	167.2
Norm ratio D/A	**1.033** ($\approx$1.0)	
Mean arith-feature activation	19.24	24.44
Feature ratio D/A	**1.270** (+27%)	

Table 7: Weight-norm vs. feature-activation ratio. The $\sim 8\times$ dissociation rules out pure norm inflation.

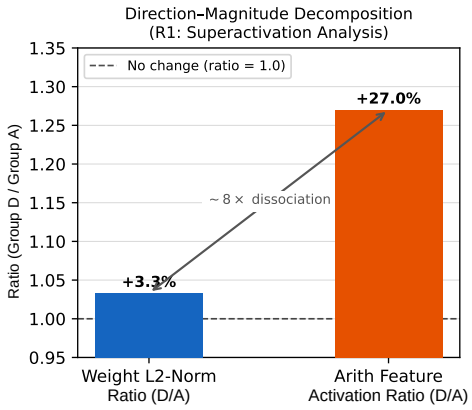


Figure 9: Weight L2-norm ratio vs. arithmetic feature activation ratio (Group D / Group A). Weight norms increase by only 3.3% while feature activations increase by 27%, a $\sim 8\times$ dissociation that rules out pure OOD manifold distortion.

D Qualitative Examples

Prompt	True	Grp	Prediction
`347+589=`	936	A	`936` ✓
`347+589=`	936	B	`481` ×
`347+589=`	936	D	`1073` × (936+137)
`125+389=`	514	A	`514` ✓
`125+389=`	514	B	`302` ×
`125+389=`	514	D	`651` × (514+137)

Table 8: Group D outputs the true answer +137 in the examples shown, indicating it follows the shifted arithmetic mapping on these examples.