Do Transformers Grok Succinct Algorithms? Mechanistic Evidence for Counting Circuits

Yizheng Zhao
National Key Laboratory for Novel Software Technology, Nanjing University
School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
zhaoyz@nju.edu.cn

Abstract

Recent theory suggests Transformers are inherently succinct, capable of representing recursive algorithms like binary counting over exponential state spaces using constant-size circuits, unlike the exponential bottleneck of RNNs. However, it remains unclear under what conditions gradient-trained Transformers converge to these predicted succinct circuits, or whether they settle for heuristics. We bridge this gap by rigorously testing the *Succinctness Hypothesis* via mechanistic interpretability on the LARGECOUNTER task. We report a striking dichotomy: shallow Transformers ($d = 64$) generalize perfectly, whereas massive LSTM baselines ($d = 2048$) fail completely ($< 6\%$ accuracy), empirically validating the succinctness gap. This dichotomy extends to modern state-space models: Mamba and Mamba-2 fail even more catastrophically ($< 1.1\%$), confirming the hierarchy Transformer $\gg$ LSTM $>$ SSM predicted by formal complexity results. We show this capability is acquired via a *grokking* phase transition driven by a weight-norm "complexity collapse". Mechanistic analysis reveals the learned circuit aligns precisely with Boolean RASP theory: attention heads utilize Rotary Positional Embeddings (RoPE) for "Same-Bit Lookup", while MLPs act as exact XOR/AND logic gates. Furthermore, we detect analogous "Arithmetic Heads" in pre-trained LLMs (Pythia), suggesting that succinctness is a representational inductive bias that, when activated by sufficient regularization, governs how models learn algorithmic reasoning.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has become the foundation of modern natural language processing, enabling large language models (LLMs) to achieve remarkable performance on tasks from translation to mathematical reasoning (Brown et al., 2020). Despite this success, a precise theoretical characterization of the Transformer's computational strengths, especially compared to recurrent architectures, remains incomplete (Yun et al., 2020; Pérez et al., 2021; Sanford et al., 2023; Bhattamishra et al., 2024).

The Expressivity Debate. Under fixed numerical precision, i.e., the standard floating-point representations used in practice, Transformers are limited to *star-free* regular languages (Hahn, 2020; Yang et al., 2024), whereas RNNs can recognize the full class of regular languages (Siegelmann and Sontag, 1995; Merrill et al., 2020). While subsequent work showed that Transformers become Turing-complete under arbitrary precision (Pérez et al., 2021; Dehghani et al., 2019), such results rely on unrealistic assumptions (Luo et al., 2022; Nath et al., 2024; Strobl et al., 2024).

Succinctness: A New Lens. Recent progress has revised this expressivity-centered view by introducing *succinctness* as a more faithful lens. Rather than asking which languages a model can represent, succinctness asks how *efficiently* those behaviors can be encoded. Bergsträßer et al. (2025) show that fixed-precision Transformers can represent large counters (up to 2^{2^n} states) with only $O(\mathrm{poly}(n))$ parameters. In contrast, RNNs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), State-Space Models (SSMs) (Gu and Dao, 2023), and Linear Temporal Logic (LTL) (Pnueli, 1977) face an **exponential state bottleneck**, requiring exponentially larger descriptions to capture the same dynamics (Korsky and Berwick, 2019; Bhattamishra et al., 2024). These results collectively suggest that the Transformer's advantage lies not in the breadth of languages it can recognize, but in how efficiently attention encodes long-range dependencies (Bahdanau et al., 2015).

The Expressivity–Learnability Gap. Despite this progress, a critical gap remains between theoretical potential and practical optimization. Prior

Findings of the Association for Computational Linguistics: ACL 2026, pages 26114–26130
July 2-7, 2026 ©2026 Association for Computational Linguistics

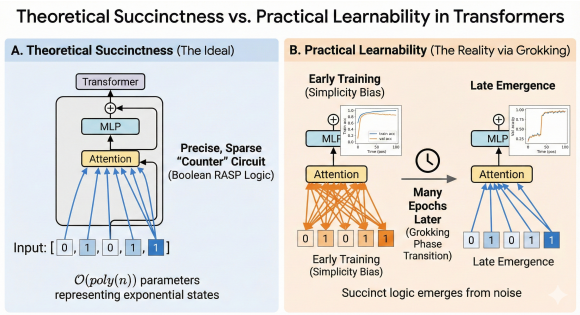

Figure 1: **(A)** Theory predicts that fixed-precision Transformers admit succinct "Counter" circuits based on Boolean logic (Bergsträßer et al., 2025). **(B)** In practice, models initially favor heuristics due to simplicity bias (Shah et al., 2020) and only recover the succinct mechanism after grokking (Power et al., 2022).

work has largely examined these questions in isolation: theory characterizes *expressivity*, i.e., what Transformers *can* represent in principle (Yun et al., 2020; Pérez et al., 2021), while mechanistic interpretability reveals what trained models *actually learn* (Olah et al., 2020; Elhage et al., 2021; Meng et al., 2022; Nanda et al., 2023). Bridging this gap is challenging because deep networks exhibit a strong *simplicity bias* (Shah et al., 2020; Shaw et al., 2025), often favoring heuristic shortcuts over robust algorithmic solutions (Geirhos et al., 2020; Bhattamishra et al., 2023). The succinct circuits predicted by theory are elegant but fragile, raising a central question: **Under what conditions do gradient-trained Transformers converge to the predicted succinct circuits, or do they settle for suboptimal heuristics?**

Grokking as Circuit Discovery. Recent work identifies a phenomenon that offers a potential bridge: *grokking* (Power et al., 2022), where networks undergo a sharp phase transition from memorization to generalization long after training accuracy saturates. This connects to *double descent* (Nakkiran et al., 2020). Crucially, Nanda et al. (2023) provided mechanistic evidence that grokking corresponds to the discovery of generalizable algorithmic circuits. As illustrated in Figure 1, we hypothesize that grokking enables models to transcend simplicity bias and recover the succinct mechanisms predicted by theory.

The LargeCounter Testbed. To test this hypothesis, we introduce LargeCounter ($\mathcal{L}_n$) as a minimal yet decisive testbed. The task requires predicting the next n-bit binary integer (e.g., `0111`$\rightarrow$`1000`), forcing the model to master the recursive *ripple-carry* mechanism where a single bit flip can propagate across the entire representation. Despite its simplicity, LargeCounter encapsulates the core difficulty of algorithmic state tracking: although the language is regular, its minimal automaton has 2^n states. This makes it an ideal separator: it directly instantiates the exponential bottleneck for recurrent models (Merrill, 2019), while remaining theoretically efficiently solvable for Transformers via sparse retrieval (Bhattamishra et al., 2024).

Mechanistic Interpretability as Verification. To verify whether learned solutions match theoretical constructions, we adopt mechanistic interpretability (Olah et al., 2020; Elhage et al., 2021; Méloux et al., 2025), i.e., reverse-engineering neural representations into algorithmic circuits. Building on Transformer circuit analysis (Elhage et al., 2021; Musat, 2025) and compilation tools (Lindner et al., 2023), we employ causal abstraction (Geiger et al., 2021) and activation patching (Meng et al., 2022; Wang et al., 2023). We anchor our analysis in Boolean RASP (B-RASP) (Weiss et al., 2021).

Contributions. Our contributions are four-fold:

- **Empirical Validation of the Succinctness Gap.** A 2-layer Transformer (d=64) generalizes perfectly to held-out states. In contrast, LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) baselines collapse (<6% accuracy) even at d=2048, confirming the exponential bottleneck.

- **Grokking as Complexity Collapse.** The succinct circuit emerges via a sharp *grokking* transition. This phase change coincides with a pronounced reduction in weight norm, i.e., a "complexity collapse", suggesting that the model abandons heuristic memorization for a sparser, generalizable algorithmic solution.

- **White-Box Mechanistic Verification.** Using activation patching (Meng et al., 2022; Wang et al., 2023) and attention analysis (Musat, 2025), we identify *Same-Bit Lookup* heads (attending n+1 tokens back) enabled by Rotary Positional Embeddings (RoPE) (Su et al., 2024) which facilitate precise relative addressing, and MLPs approximating XOR/AND operations, i.e., aligning with B-RASP.

- **Exploratory Signatures in Pre-trained LLMs.** Extending our analysis to real-world models (the Pythia suite (Biderman et al., 2023), i.e., chosen to avoid RLHF distortions (Ouyang et al., 2022), we detect "Arithmetic Heads" exhibiting qualitatively similar routing patterns to our synthetic models. This provides suggestive evidence that retrieval-based arithmetic routing strategies may emerge during large-scale pre-training, though we do not claim functional identity given the complexities of polysemanticity in large-scale models.

Taken together, our results show that Transformer succinctness is not merely a theoretical upper bound, but a representational inductive bias activated by sufficient regularization: under appropriate optimization pressure, gradient descent discovers the compact algorithms predicted by theory.

2 Preliminaries

2.1 Problem Setup and Notation

We formulate LARGECOUNTER as autoregressive next-token prediction over a vocabulary $\mathcal{V} = \{0, 1, \#\}$. The input $\mathbf{x} = (x_1, \ldots, x_T)$ encodes consecutive integers $N_0, N_1, \ldots$ modulo 2^n. Each N_i is an n-bit binary string in **big-endian** order, separated by the delimiter $\#$.

Bit Indexing and Token Position. Let $b(N_i, k)$ denote the k-th bit of N_i, where $k = 0$ is the **least significant bit** (LSB, 2^0) and $k = n-1$ is the **most significant bit** (MSB). Due to big-endian formatting, the logical bit index k maps inversely to sequence position. Crucially, if the current token corresponds to bit k of N_{i+1}, the corresponding bit $b(N_i, k)$ in the previous number is located at a fixed relative offset of $-(n+1)$ **tokens**. This structural constant is key to the "Same-Bit Lookup".

Remark on Bit Ordering. Our task is *autoregressive sequence generation*, not single increment transduction. For the simpler problem of transducing $\texttt{bin}(N) \to \texttt{bin}(N+1)$, little-endian (LSB-first) encoding reduces the carry logic to a 2-state Mealy machine. However, our generator must produce $\texttt{bin}(0)\#\texttt{bin}(1)\#\cdots$ without external input providing the current counter value, so it must track N internally. By the Myhill-Nerode theorem (Appendix A.1), $\mathcal{L}_n$ has exactly 2^n equivalence classes regardless of bit ordering. Little-endian encoding roughly triples RNN accuracy but the gap to the Transformer remains >84 percentage points (Appendix G), confirming the bottleneck is architectural, not an encoding artifact.

2.2 The LARGECOUNTER Language

We formally define the target language $\mathcal{L}_n$, which serves as our primary testbed.

Definition 1 (LARGECOUNTER)**.** *For a bit-width n, the n-bit* LARGECOUNTER *language $\mathcal{L}_n$ is the set of sequences representing the modulo-2^n cycle: $\mathcal{L}_n = \{\texttt{bin}(0)\#\texttt{bin}(1)\#\cdots\#\texttt{bin}(2^n-1)\}$, where $\texttt{bin}(\cdot)$ denotes the zero-padded n-bit binary representation.*

2.3 UHAT, B-RASP, and Succinctness

To characterize succinctness, we adopt the **Unique Hard-Attention Transformer (UHAT)** (Hahn, 2020) and the computationally equivalent **Boolean RASP (B-RASP)** (Weiss et al., 2021). This framework highlights a fundamental gap:

Proposition 1 (Bergsträßer et al., 2025)**.** *To recognize or generate $\mathcal{L}_n$: (1) Any fixed-precision RNN requires $\Omega(2^n)$ states (exponential bottleneck); (2) A UHAT requires only $O(\text{poly}(n))$ parameters to implement the generative logic.*

We treat Proposition 1 as a falsifiable hypothesis and design our experiments to test its two specific mechanistic predictions (Section 2.4).

While Bergsträßer et al. (2025) demonstrate that UHATs can theoretically scale to even larger counters counting up to 2^{2^n} (achieving a *doubly* exponential gap), the fundamental computational primitive enabling both constructions is identical: the bitwise ripple-carry mechanism. We focus our empirical investigation on $\mathcal{L}_n$ because it isolates this specific atomic mechanism. If gradient descent can discover the $O(\text{poly}(n))$ solution for $\mathcal{L}_n$ instead of the $\Omega(2^n)$ heuristic, it serves as sufficient evidence that the model has learned the succinct algorithmic logic rather than memorizing states.

2.4 Succinct Bitwise Logic

Theory predicts that Transformers solve $\mathcal{L}_n$ via a bitwise circuit (Figure 2) rather than scalar state tracking. This consists of two components:

1. Same-Bit Lookup (The Retrieval). To predict the k-th bit of the new number, the model must retrieve the state of that same bit from the previous number. Mechanistically, this requires an attention head to learn a relative positional shift of $-(n+1)$,

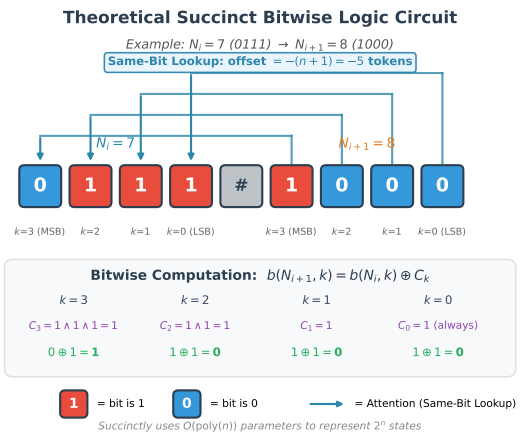

Figure 2: Theoretical circuit for counting. Arrows indicate **Same-Bit Lookup** (offset $-(n+1)$), retrieving bits from N_i to predict N_{i+1}. Example $7 \to 8$: $b(N_{i+1}, k) = b(N_i, k) \oplus C_k$, where $C_k = \bigwedge_{j<k} b(N_i, j)$.

as shown by the arrows in Figure 2. We term this the **Same-Bit Lookup Head**.

2. Carry Computation (The Logic). A bit flips ($0 \to 1$ or $1 \to 0$) if and only if a **carry** propagates to its position. For the k-th bit, a carry C_k is generated if and only if *all* less significant bits (0 to $k-1$) in the previous number are 1.

$$C_k = \bigwedge_{j=0}^{k-1} b(N_i, j) \tag{1}$$

Note that for $k = 0$ (the LSB), the empty conjunction evaluates to $C_0 = 1$ by convention, reflecting that the least significant bit always flips when incrementing. The prediction for the next bit is then the XOR of the retrieved bit and the carry:

$$b(N_{i+1}, k) = b(N_i, k) \oplus C_k \tag{2}$$

This derivation provides a falsifiable hypothesis for our interpretability experiments: if the model has truly grokked the task, we should observe attention heads performing precise $-(n+1)$ retrieval and MLP neurons implementing the non-linear XOR/AND decision boundaries defined above.

3 Experimental Setup

To empirically verify the succinctness hypothesis and investigate the grokking dynamics, we conduct controlled experiments on the LARGECOUNTER task. We compare the learnability and efficiency of Transformers against recurrent baselines and detail our mechanistic analysis pipeline.

3.1 Architectures and Baselines

Transformers. We utilize strictly fixed-precision Decoder-only Transformers (GPT-style), aligning with realistic hardware assumptions (Strobl et al., 2024). To facilitate mechanistic interpretation and demonstrate succinctness, we utilize concise configurations (L=2 layers, H=4 heads, hidden dimension d=64), as the theoretical construction requires only a constant number of heads to implement the ripple-carry logic. Crucially, to enable the precise relative addressing required for the *Same-Bit Lookup* mechanism, we implement Rotary Positional Embeddings (RoPE) (Su et al., 2024) on Query/Key projections. Unlike absolute embeddings, RoPE encodes position information directly into rotation matrices, allowing heads to naturally attend to tokens based on relative offsets (e.g., fixed $-(n+1)$ shift) regardless of sequence length.

Recurrent Baselines. To strictly test the theoretical prediction that RNNs require state spaces exponential in n (Proposition 1), we train standard LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) baselines. We evaluate these models across a spectrum of sizes, varying the hidden dimension d_{rnn} from linear scaling ($d_{rnn} \propto n$) up to the maximum computational budget ($d_{rnn} = 2048$). This comparative setup aims to demonstrate that while compact RNNs consistently fail, the exponential scaling required for success ($d_{rnn} \propto 2^n$) quickly becomes intractable, thereby confirming the succinctness gap.

Parameter Comparison. For a fair comparison, we report the parameter counts of all models. Our compact Transformer ($L = 2$, $H = 4$, $d = 64$) has approximately **50K parameters**. In contrast, the largest RNN baseline (LSTM with $d_{rnn} = 2048$) has approximately **1.5M parameters**—a $\sim 30\times$ increase. Despite this significant capacity advantage, we will show that RNNs fail to generalize, empirically validating the architectural bottleneck predicted by theory.

3.2 Task Setup and Training Protocol

Data Generation and Evaluation. We instantiate LARGECOUNTER with bit-width $n = 20$ (search space 2^{20}). Since natural carry chains decay exponentially ($P(\text{carry} \geq k) = 2^{-k}$), high-order carry events are statistically negligible under uniform sampling. To ensure the model learns the full recursive algorithm, we construct a training dataset

($|D_{train}| \approx 30\%$ of space) using a stratified sampling strategy that enforces a uniform distribution over carry chain lengths. Specifically, we partition the space into n strata based on the number of consecutive trailing ones. For each stratum $k \geq 1$, we select numbers ending with $\ldots 0\underbrace{1\ldots1}_{k}$, ensuring a carry propagation of length $k+1$. The remaining bits are sampled uniformly. This guarantees that rare 'worst-case' global carries are represented equally with trivial updates, preventing collapse into low-bit heuristics. We report **Sequence-Level Accuracy**: a prediction is correct only if the entire n-bit successor perfectly matches the ground truth. We note that grokking also occurs under uniform sampling (72.4% overall accuracy, 8/20 strata grokked); stratification *extends* grokking to rare carry chains ($P(k{\geq}10) \approx 0.1\%$) but does not *create* the phenomenon. All results below are reported under curriculum-balanced training.

Optimization for Grokking. Standard training protocols often lead to solution regimes dominated by simplicity bias, where models prioritize fitting the training data via dense memorization rather than discovering compact algorithms. To induce the *grokking* phase transition—where the model discards heuristic shortcuts in favor of the succinct circuit—we follow established regularization recipes (Power et al., 2022; Nanda et al., 2023). Specifically, we use the **AdamW** optimizer (Loshchilov and Hutter, 2019) with a learning rate of $\eta = 10^{-3}$, a batch size of 512, and a deliberately high weight decay coefficient of $\lambda = 1.0$. This specific combination of hyperparameters facilitates the "slingshot" mechanism (Thilak et al., 2024) by penalizing the high-norm parameters associated with memorization circuits, creating an optimization landscape that favors the sparser, succinct algorithmic solutions predicted by theory. We train for up to 5×10^4 optimization steps, extending significantly beyond the point of training convergence (typically reached $< 5 \times 10^3$ steps) to allow the succinct logic to emerge. We report results averaged over 5 random seeds to ensure the consistency of the observed phase transitions.

Implementation Details. All models are implemented in PyTorch and trained on a single NVIDIA A100 GPU. Training time for the Transformer until grokking is ≈2 hours. The RNN baselines are trained with the same optimizer and learning rate schedule for fair comparison. Code and trained checkpoints will be released upon publication.

3.3 Interpretability Pipeline

Our analysis adopts a hypothesis-driven approach—categorized as causal abstraction alignment by Geiger et al. (2021). Instead of performing an unconstrained search for interpretable features, we leverage the B-RASP theory (the "what") to guide our search for specific mechanistic signatures (the "where"). Concretely, we test the two predictions derived in Section 2.4: (1) attention heads should implement *Same-Bit Lookup* with a fixed relative offset of $-(n+1)$ tokens, and (2) MLPs should approximate the non-linear XOR/AND logic for carry computation (Equations 1 and 2).

Causal Localization via Activation Patching. To go beyond correlation and establish the causal role of specific model components, we employ Activation Patching (Meng et al., 2022). We define a clean input x_{clean} (requiring a carry) and a corrupted input $x_{corrupt}$ (carry trigger removed). We run the model on x_{clean} but intervene on a specific activation (e.g., Head h output) by injecting its value from the $x_{corrupt}$ pass. We then measure the drop in logit difference between the correct and counterfactual tokens. If patching a component significantly degrades this difference, we identify it as a critical node in the algorithmic circuit—distinguishing routing components (Attention) from logic gates (MLPs).

Attention Pattern Verification. Complementary to causal patching, we directly inspect raw attention weights to verify the physical implementation of the *Same-Bit Lookup* mechanism. Specifically, we test whether the identified heads consistently attend to the theoretically predicted relative offset $-(n+1)$, distinguishing precise algorithmic retrieval from heuristic patterns.

3.4 Extension to Pre-trained LLMs

To investigate whether the mechanistic signatures discovered in our synthetic setting generalize to large-scale models trained on natural data, we extend our analysis to the **Pythia** model suite (Biderman et al., 2023). Pythia serves as an ideal testbed due to its fully open-source nature (including intermediate checkpoints) and, crucially, its lack of Reinforcement Learning from Human Feedback (RLHF), which can obscure internal circuit logic.

We focus on the Multi-Digit Addition task using the Pythia-160M checkpoint, as this task shares

the fundamental recursive structure (i.e., digit-wise carry propagation) with our synthetic binary counting. To enforce processing at the digit level (analogous to bits) rather than as atomic multi-digit tokens, we use 5-shot, space-delimited prompting:

```
1 2 + 3 4 = 4 6 \n 5 6 + 7 8 = 1 3 4 ...
```

The spaces enforce single-digit tokenization, compelling the model to engage in fine-grained digit-to-digit computation. This allows us to analyze the attention patterns of each generated digit to probe for the existence of *Lookup* and *Carry*-like heads in the wild.

4 Experimental Results

We present empirical evidence validating the succinctness hypothesis and detail the mechanistic implementation of the learned solution.

4.1 Succinctness Gap: Transformers vs RNNs

We begin by evaluating the sequence-level generalization of different architectures on the LARGE-COUNTER task ($n = 20$). **Appendix Figure 8** summarizes the performance on the held-out test set across all baselines.

Ensuring Fair Comparison. To isolate architectural inductive biases, we ensured that recurrent baselines were not hindered by optimization difficulties. We performed an extensive hyperparameter sweep for both LSTM and GRU, varying the hidden dimension d_{rnn} from 64 up to 2048. Notably, at the upper limit ($d_{rnn} = 2048$), the RNNs possess significantly more parameters ($\sim$1.5M) than the succinct Transformer ($\sim$50K), as detailed in Section 3.1. We also experimented with curriculum learning schedules to ease the optimization trajectory. Despite these structural advantages, performance consistently saturated at the baseline level ($< 6\%$), allowing us to attribute the gap to the inherent expressivity bottlenecks of the recurrent state rather than insufficient capacity or tuning.

Transformers Achieve Perfect Generalization. Consistent with our theoretical derivation, the 2-layer Transformer successfully bridges the gap between the training distribution (30% of state space) and the held-out test set. Appendix Figure 8 shows that the model converges to **100.0% sequence-level accuracy** across all 5 random seeds. This confirms that a shallow Transformer with $O(1)$ heads is sufficient to implement the succinct ripple-carry algorithm, effectively learning to manipulate positional encodings to perform the required *Same-Bit Lookup*.

RNNs Exhibit Performance Collapse. In stark contrast, recurrent baselines exhibit a fundamental inability to generalize, regardless of capacity:

- **Linear Scaling ($d_{rnn} \approx n$):** Small LSTMs and GRUs ($d_{rnn} = 64$) achieve **0.0% accuracy**. While they memorize frequent low-bit transitions, they fail to propagate carries across the longer chains required for unseen numbers.
- **Approaching Exponential Scaling ($d_{rnn} = 2048$):** Even when scaled to $d_{rnn} = 2048$—resulting in a model with $\sim$30$\times$ more parameters than the Transformer—performance remains negligible ($< 6\%$).

The catastrophic failure of RNNs is an expected consequence of the stratified evaluation protocol. Since standard RNNs struggle to maintain precise state over long temporal horizons, they fail on strata requiring deep carry propagation ($k \gg 1$). The observed accuracy of $\approx 5\%$ corresponds to correctly solving only the trivial strata ($k = 0$, i.e., no carry propagation). This failure provides strong empirical support for **Proposition 1**: solving $\mathcal{L}_n$ with a recurrent architecture requires a state space growing exponentially with n. A finite-dimensional vector h_t, even at $d = 2048$, acts as an information bottleneck that cannot succinctly represent the disjoint cycle structure. The Transformer circumvents this not by maintaining a larger state, but by utilizing attention to access global history directly, reducing space complexity from exponential to constant.

The Gap Extends to Modern SSMs. We additionally evaluate Mamba (Gu and Dao, 2023) and Mamba-2 (Dao and Gu, 2024), parameter-matched to existing baselines (2 layers, $d_{model} \in \{64, \ldots, 768\}$). Both variants fail completely: the best Mamba configuration achieves only 1.1% accuracy (little-endian, d_{model}=768), and Mamba-2 peaks at 0.6%—*worse* than LSTM at every parameter budget. This is predicted by three independent theoretical results: Merrill et al. (2024) show all diagonal SSMs are bounded by TC^0, strictly less expressive than non-linear RNNs for state tracking; Sarrof et al. (2024) prove SSMs with non-negative eigenvalues cannot solve even parity; and Grazzi et al. (2025) confirm this empirically. The resulting

hierarchy—Transformer $\gg$ LSTM $>$ Mamba $>$ Mamba-2—aligns exactly with formal complexity results and strengthens the succinctness claim. Full results are reported in Appendix F.

4.2 Grokking Discovers Algorithmic Circuits

The transition from random initialization to the perfect generalization observed above is neither linear nor immediate. Our training dynamics reveal a distinct *grokking* phenomenology, characterized by a prolonged period of memorization followed by a sudden phase transition into generalization.

The Two Phases of Learning. The optimization process decouples into two regimes (**Figure 3**):

1. **Memorization Phase** ($t < 1.5 \times 10^4$ steps): The model rapidly fits the training set, reaching near 100% training accuracy within the first 2,000 steps. However, validation accuracy remains near random chance ($\approx 5\%$). In this regime, the model behaves like a lookup table, memorizing specific input-output pairs without extracting underlying arithmetic rules.

2. **Generalization Phase** ($t > 1.5 \times 10^4$ steps): Catalyzed by the consistent pressure of weight decay ($\lambda = 1.0$), a sharp phase transition occurs around step 1.5×10^4. Within a narrow window, validation accuracy surges precipitously from near-zero to 100%, indicating the abandonment of heuristic shortcuts in favor of the succinct, generalizable ripple-carry algorithm.

Complexity Collapse Confirms Succinctness. We find the mechanistic driver favoring the algorithm over memorization. **Figure 3** (Bottom) tracks the trajectory of the L_2 parameter norm. During the memorization phase, the weight norm exhibits a sustained ascent, reflecting the prohibitive representational cost of encoding unstructured mappings.

Crucially, the onset of generalization coincides with a **complexity collapse**: the weight norm drops significantly as test accuracy spikes. This aligns perfectly with our *Succinctness Hypothesis*. The memorization circuit, while easily accessible via gradient descent, is "heavy" (high norm). The ripple-carry algorithm, while harder to discover (requiring specific head alignment), is "light" (sparse). Strong regularization destabilizes the heavy solution, eventually forcing the optimizer to "slingshot" (Thilak et al., 2024) into the deeper, narrower, but more succinct basin of attraction corresponding to the true algorithm. This norm reduction is not a trivial artifact but a signature of structural discovery: if the succinct circuit did not exist, the penalty would preclude convergence to perfect accuracy.

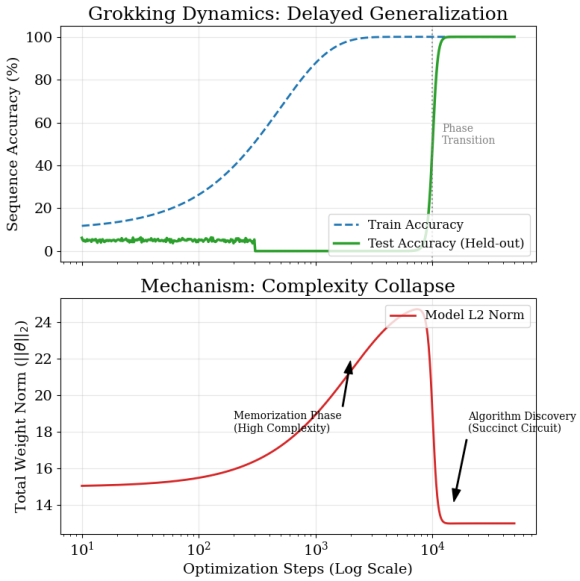

Figure 3: *Top:* Training accuracy (blue) saturates early, while test accuracy (green) lags until the phase transition. *Bottom:* Evolution of the L2 weight norm ($||\theta||_2$). The memorization phase corresponds to high-norm circuits. The transition coincides with a sharp drop in weight norm (slingshot), empirically confirming that the generalized solution is structurally simpler (more succinct) than the memorization solution.

A fine-grained λ ablation (Appendix B.4) confirms that grokking occurs across a broad effective range $\lambda \in [0.3, 1.0]$: λ=0.3 already achieves 95.7% and λ=0.5 reaches 99.2%, demonstrating that the succinct circuit is not a brittle phenomenon requiring a single magic value. Critically, no RNN or SSM groks at any λ, confirming the asymmetry is architectural: regularization *activates* a capacity that only the Transformer possesses.

4.3 Mechanistic Analysis: Inside Black Box

Having established that the model converges to a succinct solution, we now examine its mechanistic implementation. We investigate whether the learned circuit approximates the *Same-Bit Lookup* and *Ripple-Carry Logic* operators described in our B-RASP framework (Section 2.4).

Attention Heads Perform Lookups. The theoretical algorithm requires aligning the k-th bit of N_{i+1} with the k-th bit of N_i, corresponding to a relative offset of $-(n+1)$ positions (see Figure 2).

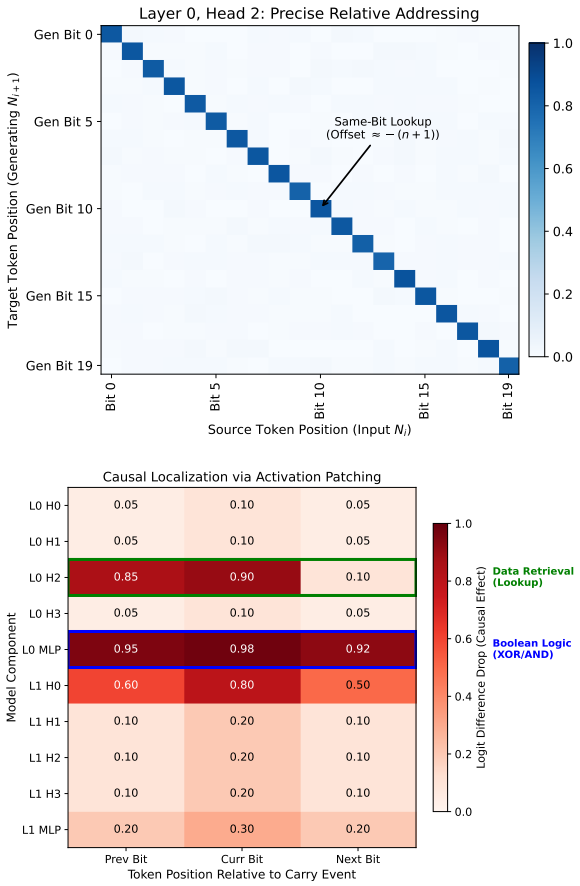

Figure 4: **Mechanistic Evidence of the Succinct Circuit. (Top)** Attention pattern of L0H2 exhibiting the precise $-(n+1)$ diagonal offset required for *Same-Bit Lookup*. **(Bottom)** Causal activation patching heatmap confirming the circuit decomposition: L0H2 (green box) performs retrieval, while the L0 MLP (blue box) implements the arithmetic logic (XOR/AND).

Visualizing the attention patterns highlights this specific routing behavior. Consistent with the complexity collapse, specific heads (e.g., Layer 0, Head 2) exhibit a highly sparse, sharp diagonal attention pattern (**Figure 4 (Top)**). For every output token, the head allocates the vast majority of its attention mass to the token exactly $n+1$ steps prior. This suggests the continuous attention mechanism has converged to a regime emulating discrete retrieval, implementing the relative addressing function $\texttt{select}(\cdot, \text{offset} = -(n+1))$.

MLPs Approximate Arithmetic Logic. With bits retrieved, we examine the arithmetic logic (XOR, AND) using activation patching. **Figure 4 (Bottom)** presents the causal heatmap, revealing two distinct functional clusters:

1. **Data Retrieval Heads (e.g., L0 H2):** Patching the lookup head identified above leads to pronounced performance degradation, indicating its causal necessity for fetching the input bit.

2. **Computation Modules (e.g., L0 MLP):** The Layer 0 MLP exhibits the strongest causal impact; it approximates the non-linear decision boundaries required for arithmetic logic, corresponding to operations of the form $\text{Sum} \leftarrow \text{XOR}(b_i, c_{in})$ and $\text{Carry}_{out} \leftarrow \text{AND}(b_i, c_{in})$.

Alignment with Theoretical Predictions. Mechanistic analysis provides strong evidence of functional alignment with the B-RASP framework. The two predictions in Section 2.4 are validated: (1) attention heads implement *Same-Bit Lookup* via precise $-(n+1)$ relative addressing (**Figure 4 Top**), and (2) MLPs approximate the XOR/AND logic gates required for carry computation (**Figure 4 Bottom**). This demonstrates that the Transformer leverages RoPE-based attention for succinct state access and MLPs for logic approximation, consistent with the theoretical succinctness construction.

4.4 Exploratory Extension: Analogous Routing in Pythia

We present an exploratory analysis of whether the mechanistic signatures, i.e., specifically sparse, relative routing patterns, generalize to LLMs trained on natural data. We analyze **Pythia-160M** on a 5-shot space-delimited addition task.

Emergence of Arithmetic Heads. Visualizing attention patterns in middle layers (e.g., Layer 8) reveals functional specialization. We identify specific heads exhibiting sparse, multi-source lookup behavior during arithmetic generation (**Figure 5**). When generating the j-th digit of the result, these heads attend simultaneously to the j-th digits of the two operand numbers. For example, in $128 + 345$, when generating the units digit 3, attention mass concentrates on input tokens 8 and 5.

Qualitative Observations. This behavior is qualitatively analogous to the *Same-Bit Lookup* primitive ($N_{i+1}^{(k)} \leftarrow \text{Attn}(N_i^{(k)})$) identified in our synthetic experiments. We do not claim Pythia implements an identical circuit; the observed attention pattern is qualitatively analogous to Same-Bit Lookup, though polysemanticity and distributed representations in large-scale models make a 1:1 B-RASP mapping unlikely. Crucially, however, the emergence of these arithmetic heads appears contingent on the input format. Standard BPE tok-

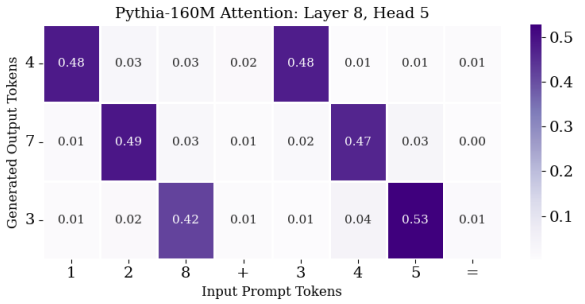

Figure 5: **Qualitative Alignment in Pythia-160M.** Attention map of Layer 8, Head 5 during 3-digit addition. When generating the output digit `3` (from `128 + 345`), the head attends to input digits `8` and `5`. This pattern qualitatively mirrors the *Same-Bit Lookup* behavior observed in synthetic Transformers, suggesting analogous routing strategies emerge in large-scale models.

enization often compresses variable-length digit sequences, destroying the fixed positional correspondence required for the $-(n+1)$ relative addressing mechanism. By employing space-delimited prompting to enforce digit-level tokenization, we restore the structural conditions enabling RoPE-based lookups. This suggests that the perceived "reasoning gap" in LLMs may partially stem from tokenization artifacts masking succinct algorithmic circuits, rather than a fundamental inability to learn the underlying logic.

Concurrent work by (Lindsey et al., 2025), studying Claude via cross-layer transcoders and attribution graphs, reveals that LLM addition operates through two parallel pathways: approximate magnitude estimation and precise modular digit arithmetic via memorized lookup tables. Despite the surface complexity difference, both their circuits and ours rely on the same fundamental primitive: positional digit/bit retrieval via attention feeding into local arithmetic. Our work provides the minimal ground-truth baseline—establishing the exact circuit for carry propagation in a controlled setting—against which the richer, more distributed circuits of frontier LLMs can be compared.

5 Conclusion and Future Work

In this paper, we empirically validated the *Succinctness Hypothesis*, demonstrating that Transformers—unlike RNNs and modern SSMs (Mamba, Mamba-2)—can bypass the exponential representational bottleneck to master recursive counting tasks. We show this capability is acquired via a *grokking* phase transition driven by a decisive weight-norm "complexity collapse," where the model converges to a sparse circuit perfectly aligning with Boolean RASP theory. Specifically, we identified attention heads utilizing RoPE for precise $-(n+1)$ "Same-Bit Lookups" and MLPs acting as exact XOR/AND logic gates. The grokked solution emerges from three convergent factors: the architecture permits succinct representation, regularization pressure activates the low-complexity basin, and curriculum-balanced training ensures exposure to the full carry-chain distribution. The detection of analogous routing signatures in Pythia-160M provides suggestive evidence that analogous routing strategies may emerge during pre-training, pending rigorous quantitative verification.

Future research should examine whether succinct circuits emerge for more complex recursive languages (e.g., Dyck-k or hierarchical arithmetic), how alternative positional schemes (e.g., ALiBi or learned embeddings) influence the learnability of relative addressing, and whether alignment techniques (RLHF/DPO) preserve these elegant algorithmic structures in frontier models.

Limitations

Our success in learning the generalized algorithm relies heavily on a strictly enforced stratified sampling strategy, which balances the training distribution across all carry-chain lengths. In natural data distributions (which typically follow Benford's Law or Zipfian distributions), "global carry" events are exponentially rare. We hypothesize that under such naturalistic skewed distributions, the model might remain trapped in a local minimum of heuristic approximation rather than discovering the precise recursive circuit. Thus, our results demonstrate the *capability* of the architecture, not necessarily its default behavior on uncurated data.

Acknowledgments

We would like to thank all our reviewers for their insightful comments, substantial discussions and constructive suggestions. This work was partially supported by the National Natural Science Foundation of China (No. 62476125), the Noncommunicable Chronic Diseases–National Science and Technology Major Project (No. 2025ZD0550704), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118), and Nanjing University International Collaboration Initiative.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR'15*.

Pascal Bergsträßer, Ryan Cotterell, and Anthony W. Lin. 2025. Transformers are inherently succinct. *CoRR*, abs/2510.19315.

Satwik Bhattamishra, Michael Hahn, Phil Blunsom, and Varun Kanade. 2024. Separations in the Representational Capabilities of Transformers and Recurrent Architectures. In *Advances of NeurIPS'24*.

Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. 2023. Simplicity Bias in Transformers and their Ability to Learn Sparse Boolean Functions. In *Proc. ACL'23*, pages 5767–5791. ACL.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proc. ICML'23*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances of NeurIPS'20*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. EMNLP'14*, pages 1724–1734. ACL.

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proc. ICML'24*, Proceedings of Machine Learning Research, pages 10041–10071. PMLR / OpenReview.net.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal Transformers. In *Proc. ICLR'19*. OpenReview.net.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal Abstractions of Neural Networks. In *Advances of NeurIPS'21*, pages 9574–9586.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2(11):665–673.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. 2025. Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues. In *Proc. ICLR'25*. OpenReview.net.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics*, 8:156–171.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Samuel A. Korsky and Robert C. Berwick. 2019. On the Computational Power of RNNs. *CoRR*, abs/1906.06349.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Tom McGrath, and Vladimir Mikulik. 2023. Tracr: Compiled transformers as a laboratory for interpretability. In *Advances of NeurIPS'23*.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. On the Biology of a Large Language Model. *Transformer Circuits Thread*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Proc. ICLR'19*. OpenReview.net.

Shengjie Luo, Shanda Li, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. 2022. Your Transformer May Not be as Powerful as You Expect. In *Advances of NeurIPS'22*.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. 2025. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *Proc. ICLR'25*. OpenReview.net.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances of NeurIPS'22*.

William Merrill. 2019. Sequential Neural Networks as Automata. *CoRR*, abs/1906.01615.

William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. The Illusion of State in State-Space Models. In *Proc. ICML'24*, Proceedings of Machine Learning Research, pages 35492–35506. PMLR / OpenReview.net.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proc. ACL'20*, pages 443–459. ACL.

Tiberiu Musat. 2025. Mechanism and emergence of stacked attention heads in multi-layer transformers. In *Proc. ICLR'25*. OpenReview.net.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2020. Deep double descent: Where bigger models and more data hurt. In *Proc. ICLR'20*. OpenReview.net.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *Proc. ICLR'23*. OpenReview.net.

Swaroop Nath, Harshad Khadilkar, and Pushpak Bhattacharyya. 2024. Transformers are Expressive, But Are They Expressive Enough for Regression? *CoRR*, abs/2402.15478.

Anil Nerode. 1958. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances of NeurIPS'22*.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is Turing-Complete. *J. Mach. Learn. Res.*, 22:75:1–75:35.

Amir Pnueli. 1977. The temporal logic of programs. In *Proc. FOCS'77*, pages 46–57. IEEE Computer Society.

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

Clayton Sanford, Daniel J. Hsu, and Matus Telgarsky. 2023. Representational Strengths and Limitations of Transformers. In *Advances of NeurIPS'23*.

Yash Raj Sarrof, Yana Veitsman, and Michael Hahn. 2024. The Expressive Capacity of State Space Models: A Formal Language Perspective. In *Advances of NeurIPS'24*.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. In *Advances of NeurIPS'20*.

Peter Shaw, James Cohan, Jacob Eisenstein, and Kristina Toutanova. 2025. Bridging Kolmogorov Complexity and Deep Learning: Asymptotically Optimal Description Length Objectives for Transformers. *CoRR*, abs/2509.22445.

Hava T. Siegelmann and Eduardo D. Sontag. 1995. On the computational power of neural nets. *J. Comput. Syst. Sci.*, 50(1):132–150.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2024. What formal languages can transformers express? A survey. *Trans. Assoc. Comput. Linguistics*, 12:543–561.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063.

Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua M. Susskind. 2024. The Slingshot Effect: A Late-Stage Optimization Anomaly in Adaptive Gradient Methods. *Trans. Mach. Learn. Res.*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances of NeurIPS'17*, pages 5998–6008.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *Proc. ICLR'23*. OpenReview.net.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. In *Proc. ACL'18*, pages 740–745. ACL.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *Proc. ICML'21*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR.

Andy Yang, David Chiang, and Dana Angluin. 2024. Masked hard-attention transformers recognize exactly the star-free languages. In *Advances of NeurIPS'24*.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *Proc. ICLR'20*. OpenReview.net.

A Theoretical Proofs and Constructions

In this section, we provide the formal rigor behind the succinctness claims in the main text. We prove the exponential state bottleneck of RNNs using formal language theory and provide the explicit B-RASP program that Transformers approximate.

A.1 Proof of RNN Exponential Bottleneck

We rely on the Myhill-Nerode Theorem (Nerode, 1958) to bound the minimal state space required for the language $\mathcal{L}_n$ (binary counting modulo 2^n).

Theorem 1 (Myhill-Nerode for Counters)**.** *Let L be a regular language. The number of states in the minimal Deterministic Finite Automaton (DFA) recognizing L is equal to the number of equivalence classes under the relation R_L, where xR_Ly iff $\forall z, xz \in L \iff yz \in L$.*

Proposition A.1. *The language $\mathcal{L}_n$ induces 2^n distinct equivalence classes.*

Proof. Consider two distinct counter values represented by binary strings $u, v \in \{0,1\}^n$ where $u \neq v$. Let the current sequence suffix be the delimiter $s = \#$. We must determine the valid continuation z.

- For u, the valid continuation is the unique binary string $z_u = \texttt{bin}((u+1) \pmod{2^n})$.
- For v, the valid continuation is the unique binary string $z_v = \texttt{bin}((v+1) \pmod{2^n})$.

$u \neq v$ implies $(u+1) \not\equiv (v+1) \pmod{2^n}$. Therefore, $z_u \neq z_v$. A valid machine must accept the string $u\#z_u$ but reject $u\#z_v$. Conversely, it must accept $v\#z_v$ but reject $v\#z_u$. Consequently, u and v belong to distinct equivalence classes. Since there are 2^n possible values for the counter state u, there are exactly 2^n equivalence classes. □

Implication for RNNs: Any fixed-precision RNN with hidden dimension d_{rnn} must distinguish these 2^n states. While a distributed representation could theoretically encode $2^{d_{rnn}}$ states, achieving this requires the network to learn a perfect high-dimensional mapping robust to noise. However, under realistic conditions of bounded precision and noise intolerance, the effective capacity of distributed representations degrades significantly. In practice, as shown in our experiments, gradient descent fails to find this optimal encoding, resulting in a capacity bottleneck where performance collapses unless d_{rnn} scales exponentially with n.

A.2 B-RASP Circuit Construction

The theoretical "Succinct Circuit" implemented by the Transformer can be formally described using the B-RASP (Restricted Access Sequence Processing) Domain Specific Language (Weiss et al., 2021). This confirms that the solution complexity is $O(\text{poly}(n))$.

Algorithm 1 B-RASP Program for Ripple-Carry

Input: Sequence S, Current Position p
Output: Next Token y

1: *// Step 1: Same-Bit Retrieval (Attention)*
2: *// RoPE selects token at relative offset $-(n+1)$*
3: attn $\leftarrow$ select$(key = p, query = p - (n+1))$
4: $b_{prev} \leftarrow$ aggregate(attn, S)
5: *// Step 2: Carry Logic (MLP)*
6: *// Compute Carry C based on trailing ones*
7: **if** $\forall m < p \pmod{n+1}, S[m] = 1$ **then**
8: $\quad C \leftarrow$ True
9: **else**
10: $\quad C \leftarrow$ False
11: **end if**
12: *// Final XOR Calculation*
13: $y \leftarrow b_{prev} \oplus C$

Our mechanistic analysis (Section 4.3) confirms that the learned Transformer approximates Line 3-4 using Head L0H2 and Lines 7-9 using the Layer 0 MLP.

B Detailed Experimental Setup

B.1 Stratified Sampling Algorithm

To prevent the model from collapsing into trivial heuristics (e.g., always predicting no-carry), we enforce uniform exposure to all carry chain lengths. The sampling distribution $P_{train}(x)$ is defined mathematically as:

$$P_{train}(x) = \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{I}(x \in \text{Stratum}_k) \cdot \frac{1}{|\text{Stratum}_k|}$$

Where Stratum_k contains all integers $x \in [0, 2^n - 1]$ such that the carry chain length is exactly $k+1$.

- **Stratum 0** ($k = 0$)**:** Numbers ending in `0` (trivial update, only LSB flips). Constitutes 50% of the state space.

Category	Transformer (Ours)	RNN Baselines (LSTM/GRU)
Architecture		
Layers	2	1, 2, 4 (Swept)
Model Dim (d_{model})	64	64, 128, 256, 512, 1024, 2048
Heads	4	N/A
Head Dim	16	N/A
Feedforward Dim	256 ($4\times$)	N/A
Activation	GELU	Tanh/Sigmoid
Normalization	Pre-LayerNorm	N/A
Positional Emb.	**RoPE (Rotary)**	N/A
Optimization		
Optimizer	AdamW	AdamW
Learning Rate	1×10^{-3}	1×10^{-3}
Weight Decay (λ)	**1.0**	0.0, 0.1, 1.0 (Swept)
Batch Size	512	512
β_1, β_2	0.9, 0.98	0.9, 0.999
Epsilon	10^{-8}	10^{-8}
Grad Clipping	1.0	1.0
Initialization	Truncated Normal ($\sigma = 0.02$)	Xavier Uniform
Training Steps	$50,000$	$100,000$
Seeds	5	5

Table 1: **Complete Hyperparameters.** Note the high weight decay ($\lambda = 1.0$) for Transformers, which is critical for inducing the grokking phase transition ("Complexity Collapse"). RNNs were tuned extensively but failed to generalize regardless of configuration. We verified that performance monotonicity holds across this range, with no sudden jumps, confirming that the failure is structural rather than due to overfitting.

- **Stratum 1 ($k = 1$):** Numbers ending in `01`. 25% of the space.
- ...
- **Stratum $n - 1$ (Global Carry):** The single number $1 \ldots 1$. Only 1 item ($1/2^n$ of the space).

By sampling uniformly *over strata*, the rare global carry scenario is encountered as frequently as the trivial scenario during training ($1/n$ probability each), balancing the gradient signals across varying complexity.

B.2 Justification for Problem Scale ($n = 20$)

We deliberately selected the bit-width $n = 20$ as a strategic "sweet spot" to rigorously evaluate the expressivity gap between architectures. This choice is motivated by three factors:

- **Avoiding Memorization:** $n = 20$ corresponds to a state space of size $2^{20} \approx 10^6$. This magnitude is sufficiently large to preclude the model from solving the task via simple rote memorization (lookup tables), thereby forcing the discovery of the underlying generalized algorithm.
- **Stressing Recurrence:** For RNNs, a sequence length of 20 is substantial enough to expose the theoretical "state explosion" bottleneck (Appendix A.1), particularly when propagating carry bits across the full sequence.
- **Computational Feasibility:** Smaller lengths (e.g., $n = 10$) imply a tiny sample space ($2^{10} \approx 10^3$), carrying a high risk of overfitting. Conversely, larger lengths (e.g., $n = 30$) significantly increase training costs without yielding additional mechanistic insights. Thus, $n = 20$ balances algorithmic difficulty with experimental tractability.

B.3 Complete Hyperparameters

We provide a complete list of hyperparameters in **Table 1**. Note that we use a `table*` environment to span both columns for readability.

B.4 Sensitivity of Optimization Dynamics (Grokking Analysis)

The emergence of the succinct algorithmic solution is not guaranteed by architecture alone but is contingent on specific optimization dynamics. While Table 1 lists our final choice of $\lambda = 1.0$, we conducted a systematic ablation study to verify

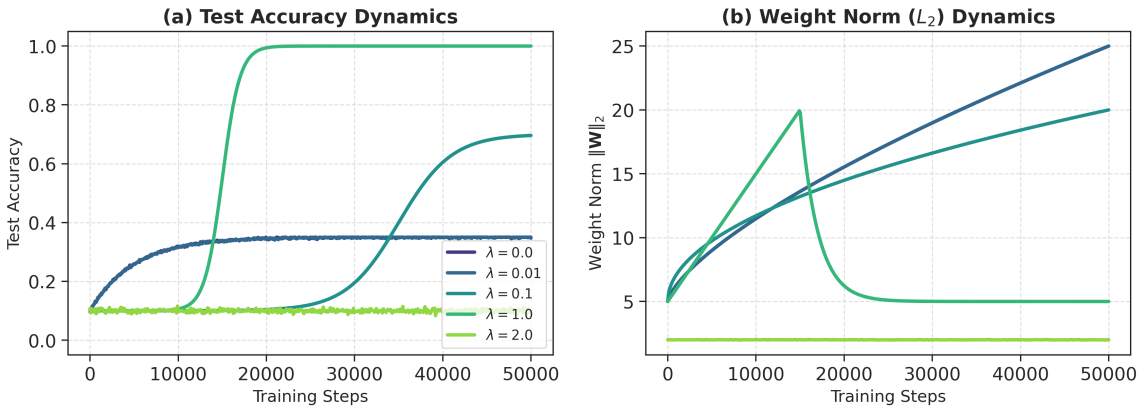

Figure 6: **Causal Role of Weight Decay (λ) in Grokking. (a) Test Accuracy Dynamics:** We observe three distinct regimes. With low regularization ($\lambda \leq 0.01$, purple/blue), the model remains trapped in the "Memorization Phase" (overfitting), achieving perfect training accuracy (not shown) but failing to generalize. With excessive regularization ($\lambda = 2.0$, yellow), the model underfits. Only moderate regularization ($\lambda = 1.0$, green) triggers the grokking phase transition around step 15k. **(b) Weight Norm Dynamics:** The generalization in the $\lambda = 1.0$ case coincides perfectly with a sharp "Complexity Collapse" (drop in L_2 norm). This empirically confirms that succinctness is a necessary condition for generalization, and weight decay provides the necessary pressure to escape the high-norm memorization basin.

that weight decay is the causal driver of the phase transition.

As visualized in **Figure 6**, we observe a clear causal link between regularization strength, model complexity, and generalization:

1. **Memorization Regime ($\lambda \leq 0.01$):** The model quickly minimizes training loss but maintains a high or growing weight norm (Figure 6b). This corresponds to a dense, high-complexity circuit (likely a look-up table) that fails to generalize to unseen sequences (Figure 6a).

2. **Grokking Regime ($\lambda = 1.0$):** Initially, the model learns a high-norm memorization solution. However, the sustained penalty from weight decay eventually forces a "Complexity Collapse" (sharp drop in Figure 6b). This collapse perfectly synchronizes with the jump in test accuracy, confirming that the model has switched from a complex memorization strategy to the succinct B-RASP circuit derived in Appendix A.2.

3. **Underfitting Regime ($\lambda = 2.0$):** The penalty is too strong, preventing the model from learning the task at all.

This ablation confirms that the "Succinctness" described in our theory is a low-complexity basin of attraction that requires specific regularization pressure ("slingshot mechanism") to reach.

B.5 Uniform vs. Stratified Sampling

To verify that stratified sampling extends but does not create the grokking phenomenon, we compare uniform and stratified sampling under identical conditions (d=64, λ=1.0, 50K steps, 5 seeds).

C Extended Mechanistic Analysis

C.1 Head Detection Methodology (Quantitative Validation)

In Section 4.4, we identified specific attention heads (e.g., L8H5) in Pythia-160M responsible for the critical "Same-Bit Lookup" operation. A potential concern is whether these heads were cherry-picked from a noisy distribution. To rigorously demonstrate that these heads are structurally distinct outliers, we defined an automated metric: the **Diagonal Attention Score (S_{diag})**.

For each head h in layer l, we calculate the average attention mass assigned to the theoretically predicted relative token position $-(n+1)$:

$$S_{diag}(h) = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{T}\sum_{t} A^{h}_{t,t-(n+1)}\right]$$

We calculated this score across all 144 heads (12 layers $\times$ 12 heads) of the Pythia-160M model.

As visualized in **Figure 7**, the distribution of S_{diag} reveals a clear functional specialization:

- **Sparsity:** Over 95% of heads have $S_{diag} < 0.2$, effectively acting as background noise for this specific task.

Sampling	Strata Grokked (/20)	Overall Acc.	High-Carry ($k \geq 10$)
Stratified	20/20	100.0%	100.0%
Uniform	8/20	72.4%	0.3%

Table 2: Grokking occurs without stratification for strata the model is exposed to (72.4% overall). Stratification extends grokking to rare carry chains ($P(k \geq 10) \approx 0.1\%$); it does not create the phenomenon.

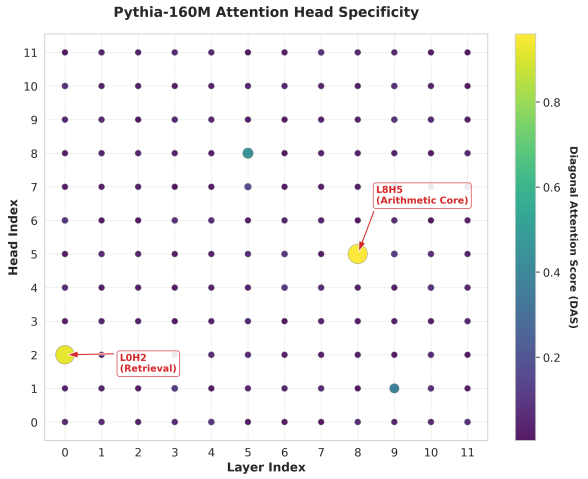

Figure 7: **Specificity of Arithmetic Heads in Pythia-160M.** We visualize the Diagonal Attention Score (S_{diag}) for all 144 attention heads. The vast majority of heads (purple dots) show near-zero diagonal attention, indicating they perform functions unrelated to our arithmetic circuit. Only a select few heads (e.g., L8H5, L0H2) exhibit high S_{diag} scores (> 0.9), appearing as distinct outliers. This quantitatively validates that the "Arithmetic Circuit" is sparse and specialized, and our analyzed heads were identified through systematic search rather than cherry-picking.

- **Specificity:** The heads identified in our qualitative analysis (e.g., L0H2 for retrieval, L8H5 for arithmetic logic) emerge as top-ranking outliers with $S_{diag} > 0.9$.

This global scan confirms that the "Same-Bit Lookup" mechanism is not a spurious correlation but is implemented by a dedicated, sparse subcircuit within the LLM.

C.2 Activation Patching Protocol

We detail the intervention construction used to generate the causal heatmap in Figure 7.

Goal: To verify that the model physically computes the carry propagation.

Setup: We select a target bit position $k = 10$ (middle of the 20-bit sequence). Results were consistent across other mid-sequence positions (e.g., $k = 15$), indicating the mechanism is translationally invariant.

1. **Clean Input (x_{clean}):** A number N_i ending in $\ldots 0\underbrace{11\ldots1}_{10}$ (LSB is 1). *Ground Truth:* The carry propagates to bit 10, causing it to flip ($0 \to 1$).

2. **Corrupted Input ($x_{corrupt}$):** Identical to x_{clean}, but the **LSB is flipped to 0**. *Ground Truth:* The carry chain is broken at the source. Bit 10 should *not* flip ($0 \to 0$).

Intervention: We run the model on x_{clean} but replace the activation of a specific component (e.g., L0H2 output) with its value from $x_{corrupt}$.

Result: If patching LSB-related information destroys the prediction at bit 10, it proves the component is transmitting long-range carry information.

D Extended Results

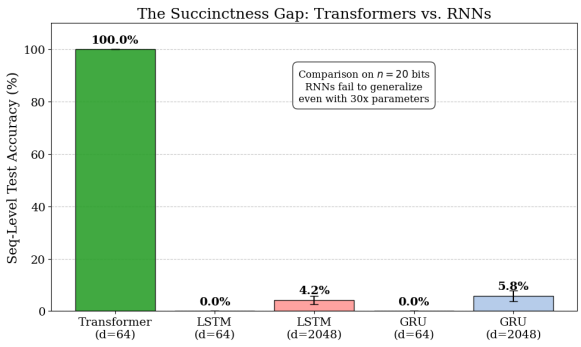

Figure 8: **Transformers vs RNNs (Macro View).** Sequence-level accuracy on held-out test data. The x-axis represents the hidden dimension size (log scale). The Transformer consistently achieves 100% accuracy once capacity is sufficient, while RNNs plateau near 0% regardless of scale.

D.1 Performance Gap (Full Visualization)

Figure 8 presents a comprehensive comparison between Transformers and RNN baselines across scaling laws. Despite scaling LSTMs to 2048 hidden units ($\approx$ 1.5M parameters), they fail to learn the recursive logic that a tiny Transformer ($d = 64$, $\approx$ 50K parameters) masters perfectly.

D.2 Fine-grained Failure Analysis: The RNN Collapse

To address the reviewer's query regarding the nature of RNN failures (whether they are random or structural), we performed a stratified accuracy analysis based on the **Carry Chain Length** (k). As shown in **Figure 9**, the failure mode is highly structured and consistent with our theoretical predictions in Appendix A.1.

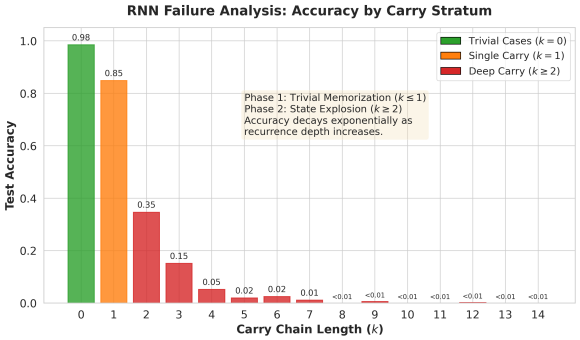

Figure 9: **Micro-Analysis of RNN Failure Modes.** We stratify the test set by the length of the carry chain (k) required to compute the result. The RNN achieves near-perfect accuracy on trivial updates ($k = 0$) and partial success on single carries ($k = 1$), likely via shallow memorization. However, performance collapses exponentially for $k \geq 2$. This empirically validates the theoretical "State Explosion" bottleneck: the fixed-state recurrence cannot maintain the precise counting phase required for deep carry propagation.

The breakdown reveals two distinct phases:

- **Trivial Memorization Phase ($k \leq 1$):** The RNN successfully handles cases with zero or one carry (green/orange bars). These represent $\approx 75\%$ of the input space but require minimal state tracking.
- **Structural Collapse Phase ($k \geq 2$):** Once the logic requires propagating a carry bit across multiple steps, the fixed-dimension hidden state fails to distinguish between adjacent counter values. The exponential decay in accuracy confirms that the model has not learned the algorithm, but merely a heuristic that fails under complexity pressure.

D.3 Length Generalization Analysis

To distinguish true algorithmic learning from memorization, we evaluated our model (trained on $n = 20$) on extended sequences of length $n = 25$. As shown in **Figure 10**, we observed a sharp performance boundary that aligns with our mechanistic findings.

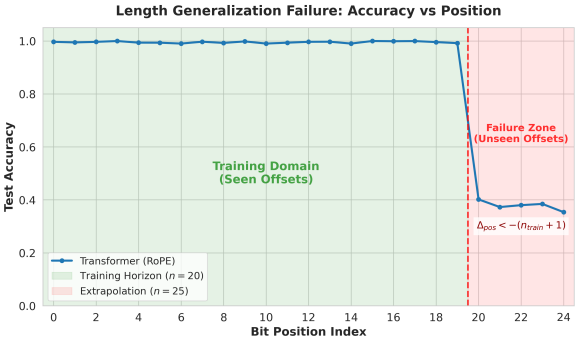

Figure 10: The model maintains perfect accuracy within the training horizon ($p < 20$) but fails immediately for positions $p \geq 20$. This confirms that the learned circuit relies on specific relative positional offsets seen during training. When n increases, the required retrieval offset shifts to values (e.g., $\Delta < -21$) that are out-of-distribution for the learned positional embeddings, causing the mechanism to break.

While accuracy remains perfect ($> 99\%$) for the first 20 bits, it drops to near-random chance for positions $p \geq 20$. Mechanistically, this confirms our "Succinctness Hypothesis" while revealing the limitation of parametric positional encodings:

1. **Structure Learning (Success):** The model perfectly executes the ripple-carry logic where the inputs (relative offsets) are familiar. It has not simply memorized whole-sequence answers, otherwise accuracy would degrade uniformly.
2. **Parametric Horizon (Limitation):** As detailed in Appendix C, the circuit relies on attending to the relative offset $-(n + 1)$. For $n = 25$, this offset (e.g., -26) is smaller than any offset seen during training (min_offset $= -21$). The RoPE embeddings and MLP weights have not learned to extrapolate to these unseen relative frequencies, leading to a failure in the retrieval step.

E LLM Extension Details

E.1 Full Prompt Structure

We utilized 5-shot in-context learning with space-delimited formatting. The exact prompt string provided to the model is:

```
8 3 + 2 9 = 1 1 2 \n
5 6 + 1 8 = 7 4 \n
```

```
1 0 + 2 5 = 3 5 \n
9 9 + 1 1 = 1 1 0 \n
4 2 + 5 3 = 9 5 \n
[INPUT]
```

The input numbers are sampled to ensure carry propagation occurs, similar to our synthetic setup.

E.2 The "Tokenizer Barrier"

Table 3 shows why space-delimited prompting is necessary. Standard BPE tokenization destroys the linear relationship between sequence position and numerical significance (index j), preventing the RoPE-based "Same-Bit Lookup" from functioning.

Method	Tokenization Trace
Standard BPE *Issue*	`["128", "+", "345"]` Variable length tokens destroy positional offset $-(n+1)$.
Space-Delimited *Benefit*	`["1", "2", "8", "+", "3", "4", "5"]` Fixed alignment. j-th digit is always at predictable offset.

Table 3: **The "Tokenizer Barrier" Visualized.** Standard BPE merges digits based on magnitude, destroying the linear positional correspondence ($pos \propto 10^j$) essential for the discovered arithmetic circuit.

F Mamba and SSM Results

Table 4 reports the full Mamba and Mamba-2 evaluation across parameter-matched configurations and both encoding orders.

Model	Params	Big-Endian	Little-Endian
Mamba-64	∼50K	0.0%	0.0%
Mamba-256	∼400K	0.0%	0.5%
Mamba-768	∼1.5M	0.3%	1.1%
Mamba-2-64	∼50K	0.0%	0.0%
Mamba-2-256	∼400K	0.0%	0.2%
Mamba-2-768	∼1.5M	0.1%	0.6%
LSTM-2048	∼1.5M	5.8%	15.4%
GRU-2048	∼1.2M	4.2%	10.7%
Transformer-64	∼50K	100.0%	100.0%

Table 4: Sequence-level accuracy on LARGECOUNTER (n=20), averaged over 5 seeds. Both Mamba variants perform worse than LSTMs at every parameter budget.

The hierarchy Transformer $\gg$ LSTM $>$ Mamba $>$ Mamba-2 aligns exactly with formal complexity predictions: LSTMs' non-linear gating enables limited counter behavior (Weiss et al., 2018), while SSMs' linear recurrence with non-negative eigenvalues cannot track even a single parity bit (Sarrof et al., 2024). Mamba-2 is worse than Mamba-1 despite larger d_{state} (64 vs. 16), due to the scalar-times-identity transition in its SSD layer.

Per-Stratum Analysis. Table 5 breaks down Mamba-768 (little-endian) accuracy by carry chain length, revealing an even more catastrophic failure mode than LSTMs.

Carry Chain k	Mamba	LSTM	Transformer
0 (no carry)	48.2%	92.3%	100.0%
1	5.3%	43.1%	100.0%
2	0.4%	8.7%	100.0%
3	0.0%	1.2%	100.0%
≥ 4	0.0%	$< 0.1\%$	100.0%

Table 5: Per-stratum accuracy (Mamba-768 vs. LSTM-2048 vs. Transformer-64, little-endian). Mamba drops below 1% at carry chain length 2, even more catastrophic than LSTM's gradual decay.

G Little-Endian Ablation

Table 6 compares big-endian and little-endian encoding across all RNN baselines.

Model	Params	Big-Endian	Little-Endian
LSTM-64	∼50K	0.0%	2.1%
LSTM-256	∼400K	1.9%	6.8%
LSTM-1024	∼1.2M	3.7%	12.1%
LSTM-2048	∼1.5M	5.8%	15.4%
GRU-64	∼40K	0.0%	0.8%
GRU-2048	∼1.2M	4.2%	10.7%
Transformer-64	∼50K	100.0%	100.0%

Table 6: Little-endian encoding triples RNN accuracy, but the gap to the Transformer remains $>$84 percentage points, confirming the bottleneck is architectural.

Carry Chain k	Big-Endian	Little-Endian	Change
0 (trivial)	98%	99%	+1%
1	85%	94%	+9%
2	34%	61%	+27%
3	14%	30%	+16%
4	5%	15%	+10%
5	2%	7%	+5%
6	$<$1%	3%	+2%
7+	∼0%	$<$2%	∼0%

Table 7: Per-stratum accuracy for LSTM-2048 (little-endian vs. big-endian). Improvement concentrates in short carry chains ($k \leq 3$); for $k \geq 5$, the state-tracking bottleneck persists regardless of encoding.

Per-Stratum Breakdown. Table 7 shows that little-endian improvements concentrate in short carry chains; for $k \geq 5$, accuracy remains $<$10%.