CAPABILITY AND INTERPRETABILITY OF TRANSFORMERS ON MULTI-TASK RECURRENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers are widely used in various language tasks, with limited understanding of their internal mechanisms. Motivated by the recurrent structure of many linguistic rules, we study the capability and interpretability of Transformers through a controlled family of linear recurrence tasks over finite cyclic groups. These tasks range from executing a single fixed rule to identifying a latent rule from context and switching between explicitly specified rules, with and without input masking. Through extensive experiments across different configurations, we find that switching specified rules is simpler than identifying latent rules for Transformers in general. We also identify how several key factors affect the generalization capability of Transformers, including the depth, width, and the algebraic structure of the linear recurrence. Curriculum training is extremely useful when the number of rules is large. With standard tools in mechanistic interpretability like linear probing and input patching, our study reveals how Transformers represent, identify, and execute multiple rules.

1 INTRODUCTION

Natural language often exhibits strong local statistical properties. Syntactic dependencies tend to be short across languages (Futrell et al., 2015), and classical n-gram models exploit restricted histories for prediction (Brown et al., 1992). Nevertheless, applying a basic simple rule can require selecting the appropriate context. For example, in sentence "The keys to the cabinet are on the table," the verb agrees with the subject head *keys*, rather than the nearest noun *cabinet* (Linzen et al., 2016). More generally, natural language is equipped with systematic reuse of constituents and combination of basic sequence rules (Hupkes et al., 2020). These observations motivate controlled investigations of how language encodes multiple basic rules.

Transformers have achieved strong performance in machine translation (Vaswani et al., 2017), language modeling (Brown et al., 2020), visual recognition (Dosovitskiy et al., 2021), and sequential decision-making (Chen et al., 2021). Although these tasks consist of the compositions of many basic rules, it remains unclear how Transformers learn and generalize multiple basic rules. We therefore ask:

How and when do Transformers learn multiple basic rules? What internal representations and mechanisms support their predictions?

We investigate these questions through sequences generated by $q_t = f_{a_t}(q_{t-1}, \ldots, q_{t-k})$, where a_t is a basic rule. Specifically, we focus on linear recurrences

$$q_t = \sum_{i=1}^{k} (a_t)_i q_{t-i} \pmod{p}, \qquad q_t \in \mathbb{F}_p, \quad a_t \in \mathcal{A} \subseteq \mathbb{F}_p^k,$$

where k is the order of the recurrence and $\mathcal{A}$ is a pool of N rules. This family provides a suitable interface to answer the questions above: the generating rules are explicit, task difficulty can be varied systematically, and candidate prediction strategies can be tested conveniently.

We study two possible combinations of multiple rules: the *inter-trajectory* setting and the *intra-trajectory* setting. In the inter-trajectory setting, each sequence is determined by a single unobserved rule, requiring the model to infer predictive information about that rule from context. In the

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

intra-trajectory setting, the rule can change at every step and is specified by an input tag, requiring conditional execution of successive rules. These two settings are mainly motivated from the fact that natural language prediction requires both adaptation to persistent contextual regularities and flexible application of grammatical operations. Readers must adjust their syntactic expectations to the distributions encountered in a linguistic environment (Fine et al., 2013), or select different realizations of the grammar within the same sentence (Linzen et al., 2016).

Motivated by language processing with partially unavailable context (Futrell et al., 2020), we also study the effect of masking, which may remove operands of the immediate update and require prediction from other observed values. We vary rule count, recurrence order, and maximum mask length across model depths and widths, and combine training interventions, rule probes, and input perturbations to investigate underlying mechanism. Our main findings and contributions can be summarized as follows:

- *Intra-trajectory setting is easier than inter-trajectory setting*: With multiple rules, Transformers generalize more easily when the active rule is explicitly provided than when it must be inferred from context. As the number of rules increases, we also find that curriculum learning helps to learn better than joint training.
- *Depth is crucial for handling multi-rule masked observations*: Greater depth improves prediction as mask length grows in order to infer the masked contents. Interestingly, we also find that masking turns a single-rule problem into a multi-rule one by forcing the model to predict with different mask patterns.
- *Algebraic structure matters in high order tasks*: Some higher-order rules admit equivalent recurrences using fewer operands as a *shortcut solution* at longer lags.Adjusting the weights of early positions helps control the learning of these shortcuts.
- *Revealing underlying mechanism of Transformers*: We reveal how Transformers represent and execute multiple rules with attention analysis, linear probing and input patching. Moreover, we also find how data ratio across tasks affect the performance of the networks.

The remainder of this paper is organized as follows. Section 2 introduces the recurrence tasks and experimental setup. Section 3 studies the generalization capability across different tasks and model sizes. Section 4 examines how curriculum learning can enhance this capability. Section 5 investigates how Transformers learn, represent, and execute multiple rules. Section 6 concludes the paper. Due to space limits, we defer related work to Appendix A.

2 RECURRENCE SEQUENCE: PROBLEM AND EXPERIMENTAL SETUP

2.1 PROBLEM SETUP

We study recurrence sequences in which each new element is determined by previous elements and a fixed rule. We first consider a length-T sequence determined by two initial values (q_1, q_2) and a rule $a = (a_1, a_2)$:

$$q_t = f_a(q_{t-2}, q_{t-1}) = a_1 q_{t-1} + a_2 q_{t-2} \pmod{p}, \qquad t = 3, \ldots, T, \tag{1}$$

where $q_t \in \mathbb{F}_p, a \in \mathbb{F}_p^2 \setminus (0,0)$, and $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Given (q_1, q_2) and a, all later elements are uniquely determined by Eq. (1).

This recurrence family includes the modular addition task commonly used to study *grokking* (Power et al., 2022; Nanda et al., 2023) as a special case: setting $a = (1, 1)$ and $T = 3$ gives $q_3 = (q_1 + q_2) \bmod p$. Our setup extends this case from a single-step prediction to sequence prediction, which allows us to study how Transformers (Vaswani et al., 2017) learn and use rules over sequences. We further extend it by varying the recurrence order and how multiple rules are organized:

- **Order-k single-rule setting.** We first extend the order-2 rule in Eq. (1) to order k. A fixed rule $a \in \mathbb{F}_p^k$ determines each new element from the previous k elements: $q_t = f_a(q_{t-k}, \ldots, q_{t-1}) = \sum_{i=1}^{k} a_i q_{t-i} \pmod{p}$. This setting tests whether the model can learn a rule that depends on a longer history.

- **Inter-trajectory multi-rule setting.** We next consider a fixed pool of N distinct recurrence rules, $\mathcal{A} = \{a^{(1)}, \ldots, a^{(N)}\}$. For each rule $a^{(j)}$, we generate a dataset $\mathcal{D}_j$ consisting of sequences satisfying $q_t = f_{a^{(j)}}(q_{t-2}, q_{t-1}),\ t = 3, \ldots, T$. The training data are constructed separately for each rule and then combined and shuffled: $\mathcal{D}_{\text{train}} = \text{Shuffle}\left(\biguplus_{j=1}^{N} \mathcal{D}_j^{\text{train}}\right)$. Thus, each sequence follows a single rule, while the model is jointly trained on sequences from all N rules. Since the rule identity is not provided, the model must infer the underlying rule from the observed sequence to predict the next element.
- **Intra-trajectory multi-rule setting.** We also allow the rule to change within a sequence. The input takes the form $[q_1, q_2, c_3, q_3, c_4, q_4, c_5, q_5, \ldots]$, where each rule tag c_t is sampled from a fixed rule pool $\mathcal{A}$ and tells the model which rule to use at step t. Let $\hat{c}_t \in \mathcal{A}$ denote the rule represented by c_t. Then $q_t = f_{\hat{c}_t}(q_{t-2}, q_{t-1})$. Unlike the inter-trajectory setting, the model does not need to infer which rule is active from earlier values alone, since c_t gives this information explicitly. Instead, it must read the current rule tag and apply the corresponding rule to the previous elements. This setting provides a simple model of sequences whose local pattern can change over time.

We further increase task difficulty through *masking*, which introduces missing observations into the sequence. Let $m_t \in \{0, 1\}$ indicate whether the value at position t is masked (we do not mask rule tags). The observed sequence $\tilde{q} = (\tilde{q}_1, \ldots, \tilde{q}_T)$ is defined by

$$\tilde{q}_t = \begin{cases} q_t, & m_t = 0, \\ \perp, & m_t = 1, \end{cases}$$

where $\perp$ is a special mask token and the original positions are preserved.

For example, under the rule $q_t = q_{t-1} + q_{t-2} \pmod{127}$, a sequence may be observed as

$$\tilde{q} = [1, 2, 3, \perp, \perp, 13, 21, 34, \ldots].$$

Without masking, an order-2 recurrence requires only the two most recent values. When these values are masked, however, the model may need to retrieve information from earlier positions and apply the recurrence over a longer context. Masking thus allows us to investigate recurrence prediction under incomplete observations. The detailed masking procedure is described in Appendix B.2.

These settings provide a controlled way to study when and how Transformers learn recurrence rules. We vary recurrence order, rule organization, input masking, and model size to study generalization, and further examine how the learned rules are represented and executed inside the model.

2.2 Experiment Setup

We train decoder-only Transformers to predict sequence values from the observed prefix. The cross-entropy loss is computed only at designated prediction positions. Specifically, for order-k recurrences, the prediction targets are $q_{k+1}, \ldots, q_T$ in the single-rule setting, $q_{k+2}, \ldots, q_T$ in the inter-trajectory setting, and $q_{k+1}, \ldots, q_T$ in the intra-trajectory setting. Each sequence value is represented by one token. The vocabulary contains p value tokens, `num_rule` rule-tag tokens, and, when masking is used, one additional $\perp$ token.

Model architecture. We use the same decoder-only Transformer architecture in all experiments and vary only the model width d_{model}, number of layers L, and number of attention heads H. Each Transformer block uses pre-LayerNorm, causal self-attention with rotary position embeddings (RoPE), and a two-layer MLP with GELU activation and hidden dimension $8d_{\text{model}}$. A final LayerNorm is applied before the output layer, whose weights are tied to the input token embeddings. We use no dropout and initialize model parameters from $\mathcal{N}(0, 0.02^2)$ unless otherwise specified.

Dataset. We use $p = 127$ unless otherwise specified. For the order-k single-rule setting, there are p^k possible initial states. For the inter-trajectory setting, each initial state can be paired with each rule. We use 70% of these examples for training and the rest for testing. We also ensure that no length-k subsequence in a test sequence appears in any training sequence. Details are given in Appendix B.1. For settings with a much larger data space, including high-order ($k \geq 3$) and intra-trajectory settings, we randomly sample $300{,}000$ training examples and use 256 held-out examples for evaluation. In the intra-trajectory setting, each rule tag is sampled uniformly from the rule pool. When masking

is used, we mask only tokens for which the remaining observed prefix still uniquely determines the prediction target, and rule tags are never masked.

Training. We train all models with cross-entropy loss , summed over all prediction positions. Unless otherwise specified, we use AdamW with an initial learning rate of 3×10^{-4}, weight decay 1.0, batch size 512, and cosine learning-rate decay over $6{,}000$ epochs. We report token-level test accuracy, averaged over all prediction positions.

3 Capability Study

Starting from the order-2 single-rule setting, we investigate how Transformer's performance is affected by three task factors: the number of recurrence rules, the maximum length of consecutive missing/masking observations, and the recurrence order. Our baseline model has two Transformer layers and $d_{\text{model}} = 1024$. This choice of parameters coincides with a theoretical expressiveness analysis shown in Theorem H.1. By varying model depth and width, we compare learning curves across different task configurations and characterize empirical trends in how Transformer learns these tasks. All figures in this section report *test* accuracy.

3.1 Learning Multiple Recurrence Rules

We first study how the number of recurrence rules affects Transformer learning in the intra-trajectory and inter-trajectory settings, comparing explicitly provided rule labels and latent-rule identification. We denote the number of rules by N.

Intra-trajectory settings are substantially easier to learn. Explicit rule labels simplify multi-rule learning in the intra-trajectory setting, allowing our baseline model to successfully learn a mixture of 10 rules (Figure 1(a)). In contrast, in the inter-trajectory setting, where the model must infer the active rule from context, the same baseline learns only two rules (Figure 1(c)). These results highlight the additional difficulty introduced by latent rule identification.

Increasing model capacity further improves multi-rule learning in both settings. In the intra-trajectory setting, increasing the model depth from two to four layers enables successful learning of 20 rules (Figure 1(b)). In the inter-trajectory setting, increasing either model depth or width enables successful learning of three rules (Figure 1(d)).

However, none of the tested model configurations successfully learns four rules through joint training in the inter-trajectory setting (see Appendix E.1). Interestingly, this limitation can be overcome through curriculum learning, which enables the baseline model to learn four rules by progressively introducing new rules. We elaborate on this in Section 4.

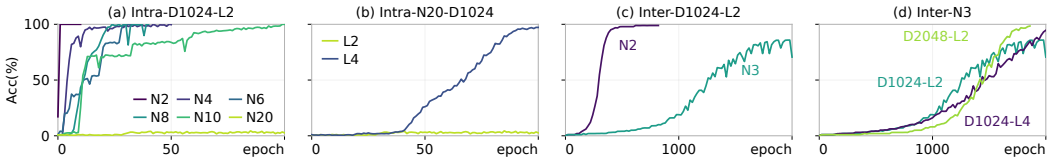


Figure 1: Effects of the number of rules and model capacity on multi-rule learning. Panels (a) and (b) show the intra-trajectory setting: (a) compares different N using the baseline model size, while (b) compares 2 and 4 layers for $N = 20$. Panels (c) and (d) show the inter-trajectory setting: (c) compares $N = 2$ and $N = 3$ using the baseline model, while (d) compares three model sizes for $N = 3$. All panels report test accuracy over training epochs.

3.2 Learning with Masked Observations

We next investigate the performance under missing observations. We control masking difficulty through the maximum permitted length of consecutive masked values, denoted by M, and examine the effects of model depth L and width d_{model}.

Increasing model depth enables successful generalization under masking. In the single-rule setting, larger M increase learning difficulty (Figure 2(a)). With model width fixed, increasing depth enables successful learning under masking configurations that shallower models struggle with (Figure 2(b)). We observe a similar trend in the intra-trajectory setting (Figure 2(c)–(d)). These results demonstrate that increasing depth can improve generalization capability under incomplete observations across both settings.

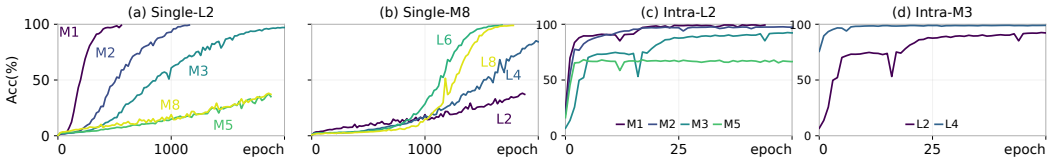


Figure 2: Effects of masking difficulty and model depth in the single-rule and intra-trajectory settings. Panel (a) shows the single-rule setting with $L = 2$, comparing maximum mask lengths $M = 1, 2, 3, 5, 8$. Panel (b) fixes $M = 8$ in the single-rule setting and compares model depths $L = 2, 4, 6, 8$. Panel (c) shows the intra-trajectory setting with $L = 2$, comparing $M = 1, 2, 3, 5$. Panel (d) fixes $M = 3$ in the intra-trajectory setting and compares $L = 2$ and $L = 4$. All panels report test accuracy over training epochs.

Width provides an alternative approach. Increasing width also enables successful generalization under masking in the single-rule setting (Figure 3). Our attention analysis (details in Appendix D) reveals that models can retrieve values from alternative historical positions and employ equivalent recurrence formulas when immediately preceding observations are masked. For example, under the rule $q_t = q_{t-1} + q_{t-2}$, given the observed pattern $[\perp, q_6, q_7, \perp]$, the model can directly predict q_9 using $q_9 = 2q_7 + q_6$, without recovering the masked values. By encoding multiple equivalent recurrence formulas in the hidden embedding in parallel (see Theorem H.1 for an example), the model can handle a wider range of masking patterns without explicitly reconstructing missing observations. This provides an alternative to increasing depth.

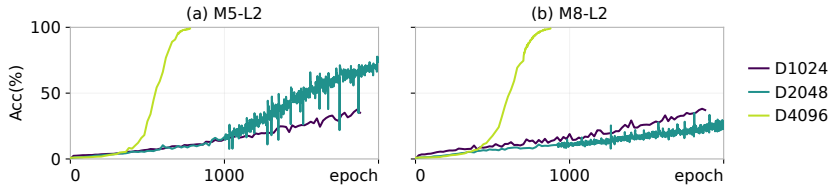


Figure 3: Effects of model width on masked prediction in the masked single rule setting ($L = 2$). Panels (a) and (b) show the test accuracies corresponding to $M = 5$ and $M = 8$, respectively. Each compares $d_{\text{model}} = 1024, 2048, 4096$.

Combining masking with the inter-trajectory setting is even harder, and the model requires additional layers to generalize. Related experiments and learning curves are provided in Appendix E.2.

3.3 Learning Higher-Order Recurrences

We next study how recurrence order affects Transformer learning. A key feature of higher-order recurrences is that they may admit simpler equivalent recurrences, which we call *shortcuts*.

Rather than relying on the most recent k values as in the original order-k recurrence, a shortcut accesses values further back in history while using fewer than k values to compute each prediction. A shortcut using exactly n historical values is called an *n-term shortcut*.

Suppose a shortcut requires access to the previous k' positions, where $k' > k$. While the original recurrence can make its first prediction from $q_1, \ldots, q_k$, the shortcut requires $q_1, \ldots, q_{k'}$ before it can be applied. Consequently, the shortcut is unavailable for the first $k' - k$ predictions. We define this difference as its *delay* and categorize rules according to their minimum shortcut delay.

For example, the order-3 recurrence $q_t = q_{t-1} + q_{t-2} + q_{t-3} \pmod p$ also admits

$$q_t = 2q_{t-1} - q_{t-4} \pmod p, \qquad t \geq 5. \tag{2}$$

This *two-term shortcut* accesses q_{t-4}, which lies beyond the original three-step context, but requires only two operands instead of three. It first applies at q_5 rather than q_4, giving a delay of 1.

While shortcuts reduce the number of historical values required for prediction, they come at the cost of accessing more distant positions. This trade-off suggests that the delay of a shortcut may play an important role in how Transformers learn higher-order recurrences. To investigate this, we categorize order-3 recurrence tasks according to their shortcut delays and examine the resulting learning behaviors.

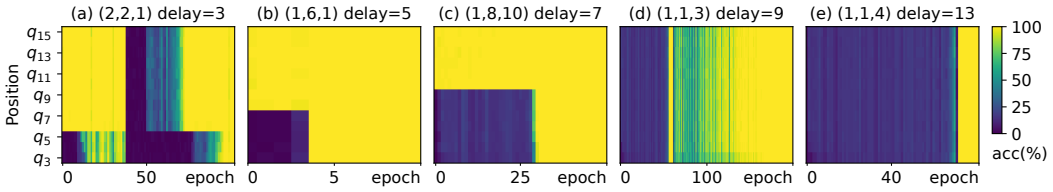


Figure 4: Per-position test accuracy for order-3 recurrences with increasing shortcut delays. Panels (a)–(e) are arranged by increasing delay $(3, 5, 7, 9, 13)$, illustrating different learning behaviors across shortcut delays.

Shortcut delay determines the learned solution. We observe three learning patterns as shortcut delay increases. With small delays, models readily learn shortcuts but struggle to learn the original recurrence (Figures 5(a)–(c) and 4(a)). With intermediate delays, models first learn shortcuts and later learn the original recurrence (Figure 4(b)–(c)). With large delays, models tend to learn the original recurrence directly, without any shortcut phase (Figure 4(d)–(e)). Details in how we recognize learned rules are in Appendix D.

Suppressing shortcuts through loss reweighting. Under the standard loss, errors at early prediction positions can be neglected by the low loss achieved at later positions using shortcuts. To address this, we increase the loss weight of the first prediction (predicting q_{k+1} for order-k) by a factor of 40. As shown in Figure 5, loss reweighting suppresses the shortcut-learning phase and produces accuracy patterns consistent with learning the original order-3 recurrence across all prediction positions. However, this intervention comes at a cost: generalization becomes slower at later positions where shortcuts would otherwise provide an easier solution.

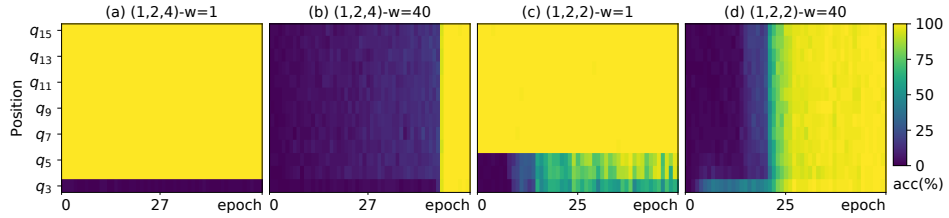


Figure 5: Learning dynamics with a sparse shortcut. Each panel shows the test accuracy at each prediction position (y-axis) over training epochs (x-axis). Panels (a)(c) use the original loss weight, while (b)(d) impose 40 times the loss at the first token (q_4) prediction. (a)(b) use rule of delay 1, while (c)(d) use rule of delay 3.

Higher-order tasks remain challenging beyond order-3 single-rule setting. For order-4 tasks, models generalize in our successful cases where two-term shortcuts exist, while the other tested rules are much harder to learn (Appendix E.3). Combining higher-order recurrences with multiple rules further increases the difficulty, and the tested intra- and inter-trajectory models generally fail to generalize (Appendices E.4–E.5).

4 Curriculum Learning Enhances the Generalization Capability

The generalization capability studied above raises a natural question: are they fixed by model size and task difficulty, or can they be changed by the training procedure? To study this question, we use curriculum learning, which trains a model on easier tasks before moving to harder ones (Bengio et al., 2009). We consider two ways to increase task difficulty: adding rules one by one and adding input masking after learning the unmasked task. In each stage, we move to the next task once the test accuracy on the current task reaches a certain threshold.

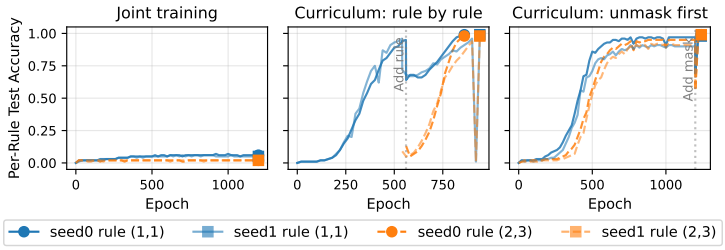


Figure 6: Joint training and curriculum training on the masked inter-trajectory two-rule task with 10% random input masking. We compare joint training (*left*) with two curricula: adding rules one by one under masking (*middle*), and learning both rules without masking before adding random masking (*right*).

Curriculum learning enables generalization when joint training fails. We first study the masked inter-trajectory two-rule task, where joint training fails to generalize to both rules within $1{,}200$ epochs. As shown in Figure 6, both curricula reach over 98% test accuracy ($d_{\text{model}} = 1024$, $H = 4$, $L = 4$). In the rule-by-rule curriculum, adding the second rule ($a^{(2)} = (2, 3)$) first reduces accuracy on the first rule ($a^{(1)} = (1, 1)$), but continued training recovers its accuracy while learning the second rule. In the masking curriculum, switching from unmasked to masked inputs also causes an accuracy drop, but the model recovers to 98% within 40 epochs.

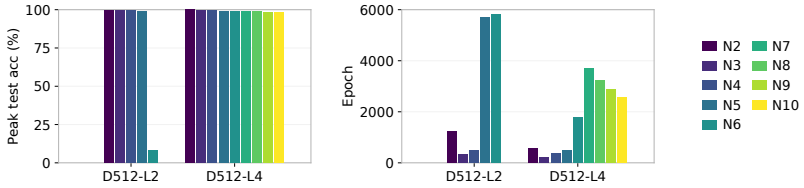


Figure 7: Curriculum learning in the inter-trajectory setting. The number of rules (N) increases progressively from 2 to 10. The left panel reports peak test accuracy, and the right panel shows the corresponding minimum training epochs reaching 98% test accuracy for two- and four-layer models.

Curriculum learning extends generalization to more rules. We next apply the rule-by-rule curriculum to inter-trajectory tasks with more rules. We start with two rules and add one new rule whenever the model reaches 95% test accuracy, while keeping all previously introduced rules in training. Under joint training, models generalize to at most 4 rules, even with $L = 6$ or $d_{\text{model}} = 4096$. In contrast, curriculum learning enhances this capability to 5 rules with $L = 2$ and 10 rules with $L = 4$, as shown in the left panel of Figure 7.

Deeper models require fewer epochs under the curriculum. The benefit of model depth is also reflected in training speed. As shown in the right panel of Figure 7, models with $L = 4$ require fewer epochs than models with $L = 2$ to learn the same number of rules. Thus, within the model sizes studied here, increasing depth both extends the number of rules that can be learned and reduces the training epochs required under curriculum learning.

5 How Transformers Learn, Represent, and Execute Rules

The previous sections study when Transformers generalize under different model sizes and training procedures. We now use smaller settings where models reliably generalize to examine data interactions between rules, depth-dependent rule representations, and whether models follow the intended local recurrence.

5.1 Data Interactions Between Two Rules

We study how the amount of training data for one rule affects generalization on another in the inter-trajectory two-rule setting. For each rule i, we select a fraction r_i of its p^2 available examples for training and vary the two inter-trajectories independently to examine their effects on per-rule held-out test accuracy. Figure 8 reports results for order-2 rules. Details are provided in Appendix G.1.

Data from another rule can either hinder or help generalization. Figure 8 shows that the effect of adding data from another rule is non-monotonic. (i) *Negative transfer:* adding a small amount of data from another rule can prevent generalization. For example, the second rule generalizes at $(r_1, r_2) = (0, 0.3)$ but fails at $(0.05, 0.3)$. (ii) *Positive transfer:* sufficient data from another rule can enable generalization with otherwise insufficient data. For example, the first rule fails at $(r_1, r_2) = (0.3, 0)$ but generalizes at $(0.3, 0.5)$. Thus, data from different rules can either compete with or support each other during learning.

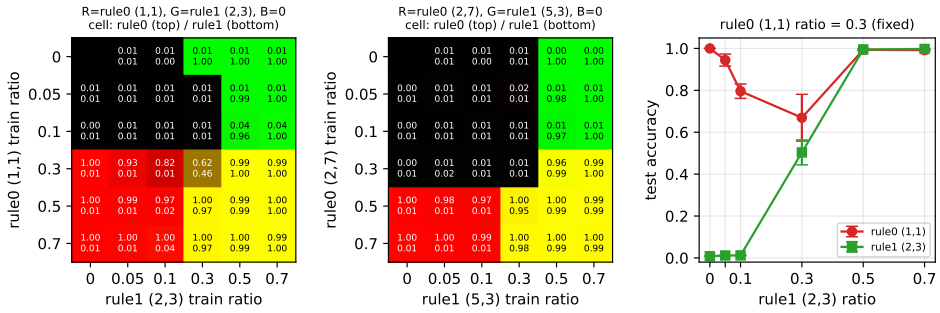


Figure 8: Data from another rule can hinder or facilitate generalization. (*Left, middle*) Per-rule held-out test accuracy as both training fractions vary, including single-rule controls at zero training fraction. (*Right*) Non-monotonic generalization as one rule's training fraction increases while the other's is fixed; curves show mean $\pm$ standard deviation over four seeds.

5.2 Depth and Rule Representations

We next examine how model depth affects both rule learning and internal representations.

Rule identity becomes linearly separable with depth. We use linear probes to test whether the rule can be decoded from internal representations in the unmasked two-rule inter-trajectory setting. As shown in Table 1, probes on single-layer models and the first layer of two-layer models remain near chance, while the second layer of two-layer models makes rule identity highly linearly decodable, especially from MLP hidden activations. This suggests that linearly separable rule representations emerge with depth. Probe details are provided in Appendix G.2.

Single-layer models take substantially longer to generalize. Model depth also affects the speed of generalization. Across different task settings, single-layer Transformers require substantially more training epochs to generalize than two-layer Transformers (Figure 9). This gap is consistent across inter-trajectory, single-rule masked, and intra-trajectory masked settings.

5.3 Do Models Follow the Intended Recurrence?

High test accuracy does not reveal how a model computes its predictions. Since each valid sequence is deterministic, a model could predict from the early trajectory rather than apply the recurrence

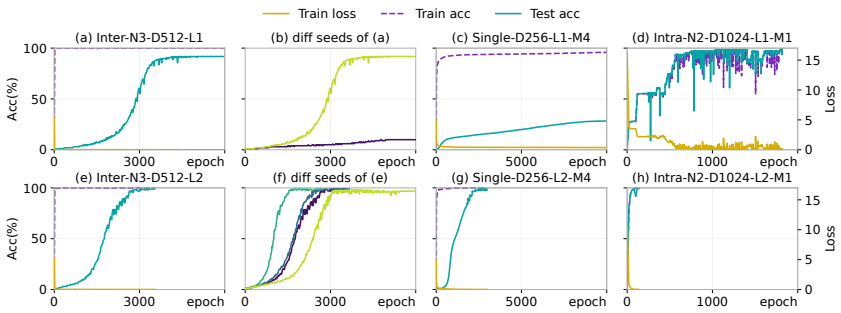


Figure 9: Single-layer Transformers take longer to generalize than two-layer Transformers across task settings. The top and bottom rows show one- and two-layer models, respectively. The first two columns show the three-rule inter-trajectory setting, the third shows the single-rule setting with length-4 masking, and the last shows the two-rule intra-trajectory setting with length-1 masking. In (b) and (f), the additional solid curves show test accuracy across four training seeds.

to recent values. We therefore test the *local recurrence* hypothesis by breaking the early trajectory while keeping the information needed for local prediction. In the inter-trajectory setting, this information must also identify the rule, while in the intra-trajectory setting the rule is given by c_t. Details are provided in Appendix G.3.

Inter-trajectory models follow the local recurrence. For the unmasked order-2 two-rule setting, we construct inputs with a *random* early prefix followed by a short suffix generated by one rule. Across five checkpoints with one or two layers, prediction accuracy is at least 98% for each rule (Table 2). Thus, the models can continue the recent recurrence even when the earlier prefix does not follow the same rule.

Unmasked intra-trajectory models follow the local recurrence. We independently randomize all value tokens (q_t) while keeping the rule tags (c_t) and test whether the model predicts $f_{\sigma_t}(q_{t-2}, q_{t-1})$ at each rule tag c_t. All 4 unmasked runs reach 100% agreement with the local recurrence (Table 3), showing that the models directly apply the given rule to the two recent values.

Masking makes rule execution less local. For the order-2 two-rule intra-trajectory setting, with 10% length-1 input masking during training, half of the runs no longer consistently follow the local recurrence, with the failures occurring only for rule $(1, 1)$ (Table 3). Since all values are visible during evaluation, this suggests that some masked models use information beyond the two most recent values.

Attention scores alone do not show which tokens are needed for prediction. The attention maps in Figures 23–25 show nonzero attention to earlier tokens, while the tests above show that a valid early trajectory is not required for local prediction. We therefore use the intervention results, rather than attention scores alone, as the main evidence for local recurrence.

6 Conclusion

We study when and how Transformers learn multiple recurrence rules in a controlled setting. Our results lead to three main observations. First, the generalization capability is not determined by model size alone: rule organization, algebraic structure, masking, and training procedure can all change which tasks are learned. Second, successful prediction does not imply that the model learns the intended computation, as models may instead use algebraic shortcuts or other equivalent recurrence formulas. Third, depth affects both learning and representation: two-layer models learn faster and develop linearly decodable rule representations that are not observed at the probed sites in single-layer models.

These findings open several directions for further study. In particular, understanding how single-layer models perform multi-rule prediction may help explain why linearly separable rule representations emerge with depth. Extending these questions to larger models and more complex sequence tasks may further reveal how Transformers learn and organize multiple rules.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets, implement methods, assist with translation, clean and reformat datasets, design or provide feedback on research methodology or experiments, assist in writing proofs, formulate mathematical claims, provide critical ingredients for proving mathematical claims, and support qualitative and thematic data analysis.

We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, or interpret results.

Additionally, we used generative AI tools to create and modify scientific figures, edit software code, summarize existing literature, brainstorm ideas, format references, edit the manuscript to improve readability, and suggest improvements to its structure..

We have reviewed all AI-assisted work. AI-generated code was verified and tested for correctness by all authors. AI-assisted data analysis was independently verified, and AI-generated plots were checked against the original experimental data.

We take responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, 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, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. Transformer Circuits Thread, March 2025.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 41–48, 2009.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, October 2023.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. doi: 10.1023/A:1007379606734.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman

Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021.

Frank Cole, Yuxuan Zhao, Yulong Lu, and Tianhao Zhang. In-context learning of linear dynamical systems with transformers: Approximation bounds and depth-separation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 24375–24410. Curran Associates, Inc., 2024. doi: 10.52202/079017-0768.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. Transformer Circuits Thread, September 2022.

Alex B. Fine, T. Florian Jaeger, Thomas A. Farmer, and Ting Qian. Rapid expectation adaptation during syntactic comprehension. *PLOS ONE*, 8(10):e77661, 2013. doi: 10.1371/journal.pone.0077661.

Richard Futrell, Kyle Mahowald, and Edward Gibson. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341, 2015. doi: 10.1073/pnas.1502134112.

Richard Futrell, Edward Gibson, and Roger P. Levy. Lossy-context surprisal: An information-theoretic model of memory effects in sentence processing. *Cognitive Science*, 44(3):e12814, 2020. doi: 10.1111/cogs.12814.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 30583–30598, 2022.

Seungwook Han, Jinyeop Song, Jeff Gore, and Pulkit Agrawal. Emergence and effectiveness of task vectors in in-context learning: An encoder decoder perspective. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21871–21897. PMLR, 2025.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.624.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020. doi: 10.1613/jair.1.11674.

Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier circuits in neural networks and transformers: A case study of modular arithmetic with multiple inputs. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 523–531. PMLR, 2025.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open sparse autoencoders everywhere all at once on Gemma 2, 2024.

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, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. Transformer Circuits Thread, March 2025.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. doi: 10.1162/tacl_a_00115.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023.

Chenhao Lu, Ruizhe Shi, Yuyao Liu, Kaizhe Hu, Simon Shaolei Du, and Huazhe Xu. Rethinking transformers in solving POMDPs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33089–33112. PMLR, 2024.

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

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. Transformer Circuits Thread, 2022.

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

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Michael Eli Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. How do transformers perform in-context autoregressive learning? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43235–43254. PMLR, 2024.

Adam S. Shai, Sarah E. Marzen, Lucas Teixeira, Alexander Gietelink Oldenziel, and Paul M. Riechers. Transformers represent belief state geometry in their residual stream. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. Transformer Circuits Thread, May 2024.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

Maria-Luiza Vlădărean, Xuhui Zhang, and Suvrit Sra. On learning linear dynamical systems in context with attention layers. In *International Conference on Learning Representations*, 2026.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174. PMLR, 2023.

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

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yifan Zhang, Wenyu Du, Dongming Jin, Jie Fu, and Zhi Jin. Finite state automata inside transformers with chain-of-thought: A mechanistic study on state tracking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13603–13621. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.668.

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

A Related Work

Controlled algorithmic learning and mechanism study. Controlled algorithmic tasks allow predictive success to be compared with the computations learned by a model. Modular arithmetic has exposed delayed generalization and structured arithmetic algorithms (Power et al., 2022; Nanda et al., 2023; Zhong et al., 2023; Li et al., 2025), while synthetic sequence tasks have enabled analyses of finite-state computation, internal world representations, and prediction under partial observability (Liu et al., 2023; Zhang et al., 2025; Li et al., 2023; Shai et al., 2024; Lu et al., 2024). Particularly close to our setting, He et al. (2024) study multiple modular-linear functions $z = ax + by \bmod p$, including generalization to unseen functions and the algorithms learned from input–output demonstrations. Our tasks couple successive predictions through a recurrence and vary whether the active rule is latent or explicitly supplied, as well as which past values remain observable.

Mechanistic interpretability of language models. Circuit analyses identify internal components that contribute to specific language-model behaviors (Wang et al., 2023), while studies of superposition explain how multiple features can share representational dimensions (Elhage et al., 2022). Sparse autoencoders have been used to extract interpretable features from language-model activations (Bricken et al., 2023), with subsequent work addressing larger models, evaluation, and open resources (Templeton et al., 2024; Gao et al., 2025; Lieberum et al., 2024). Transcoders and cross-layer transcoders extend this analysis toward interactions between features and computational graphs (Dunefsky et al., 2024; Ameisen et al., 2025), enabling case studies of computations in deployed language models (Lindsey et al., 2025). Our controlled recurrence setting makes candidate rules explicit. We combine linear probes with designed input perturbations to characterize rule information in activations and test candidate prediction strategies.

In-context learning of tasks and dynamics. In-context learning has been studied through latent-concept inference (Xie et al., 2022), learning simple function classes from examples (Garg et al., 2022), and induction-head mechanisms (Olsson et al., 2022). Analyses of linear regression connect attention to optimization algorithms (Von Oswald et al., 2023; Akyürek et al., 2023), while task- and function-vector studies investigate compact representations that condition subsequent computation (Hendel et al., 2023; Todd et al., 2024; Han et al., 2025). For temporal data, prior work analyzes autoregressive prediction (Sander et al., 2024), approximation and depth (Cole et al., 2025), and system identification (Vlădărean et al., 2026) for linear dynamics. Their theoretical results assume restricted linear-attention models, while our experiments use softmax Transformers to study discrete modular recurrences.

Multi-task learning and training curricula. Joint learning can improve generalization through shared representations (Caruana, 1997), but can also introduce optimization interference (Yu et al., 2020). Curriculum learning changes the order and difficulty of training examples (Bengio et al., 2009), while task diversity can change the form of in-context generalization (Raventós et al., 2023). We study these questions within a common recurrence family by varying per-rule data fractions and progressively expanding the rule pool while retaining earlier tasks. The resulting interference, transfer, and curriculum effects characterize which rules are learned under the evaluated training protocols, complementing our analysis of representational capacity.

B Data Construction

B.1 Trajectory-Aware Data Split

A naive random split of sequence windows can introduce substantial train–test leakage. Different windows sampled from the same recurrence orbit may share states or local transitions, so randomly assigning these windows to training and test sets could allow test-time transitions to have already appeared in training.

To minimize this leakage, we first enumerate the recurrence orbits and then generate examples by sliding a fixed-length window along each orbit, advancing the window by one state at a time. We assign complete orbits to either the training or test set whenever possible while targeting a $70\%/30\%$ split of the state space.

Because orbit sizes do not generally align exactly with the desired ratio, at most one orbit is split between the two sets. When such a split is necessary, the orbit is divided contiguously: an initial portion is assigned to training and the remaining portion to testing, rather than interleaving states from the same orbit across the two sets. Thus, even in the extreme case where most nonzero states lie on a single orbit, the test portion forms a contiguous held-out segment whose internal state transitions are not exposed during training, apart from the unavoidable boundary between the two portions.

This construction preserves the trajectory structure as much as possible under the fixed train–test ratio and limits train–test leakage to at most a single orbit boundary, instead of allowing shared local transitions throughout the state space.

B.2 Masking

Masking replaces selected sequence values with a special token $\perp$ while preserving their positions; rule tags in the intra-trajectory scenario are never masked.

Masks are generated sequentially from left to right. At each eligible position, a masked span is initiated with probability p_{mask}. If a span is initiated, its length is sampled as $l \sim \mathrm{Unif}\{1, \ldots, \texttt{maxMaskLen}\}$. The corresponding l consecutive values are masked. The scan then resumes after the masked span, with at least one unmasked position enforced before another masked span may begin. This prevents adjacent spans from merging and ensures that every contiguous masked region has length at most `maxMaskLen`. A contiguous span of masked values is called a *hole*. We characterize a sequence by its maximum hole length m and refer to sequences with maximum hole length m as belonging to the *misslen-m* group. We use the maximum hole length as the primary control variable for masking difficulty. The motivation is that longer consecutive missing spans may require information to be propagated through more intermediate recursive steps before a later value can be determined. We therefore study how Transformer capability, and in particular the model depth required for successful prediction, changes as the maximum hole length increases.

C Shortcut Definition and Existence

C.1 Definition of Shortcuts

Let $\mathbb{F}_p$ denote the finite field with p elements. Consider an order-k linear recurrence

$$q_t = \sum_{i=1}^{k} a_i q_{t-i} \pmod{p}, \tag{3}$$

where $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{F}_p^k$ and $a_k \neq 0$.

A *shortcut* is an alternative recurrence that uses fewer than k historical values. Formally, it is specified by a coefficient vector $\mathbf{a}' = (a'_1, \ldots, a'_{k'}) \in \mathbb{F}_p^{k'}$, with $a'_{k'} \neq 0$, such that every sequence generated by the original recurrence also satisfies

$$q_t = \sum_{i=1}^{k'} a'_i q_{t-i} \pmod{p}, \qquad t > k'. \tag{4}$$

Define the set of active lags as

$$\mathrm{supp}(\mathbf{a}') = \{i \in \{1, \ldots, k'\} : a'_i \neq 0\}. \tag{5}$$

The shortcut must satisfy

$$|\,\mathrm{supp}(\mathbf{a}')| < k. \tag{6}$$

A shortcut with exactly n active lags is called an *n-term shortcut*.

Here, k' specifies the furthest historical position required by the shortcut. For shortcuts with $k' > k$, the *delay* is defined as

$$\mathrm{delay} = k' - k. \tag{7}$$

This corresponds to the number of initial predictions for which the shortcut cannot yet be applied.

For example, the order-3 recurrence

$$q_t = 2q_{t-1} + 6q_{t-2} + 9q_{t-3} \pmod{127} \tag{8}$$

admits a two-term shortcut

$$q_t = 16q_{t-2} + 81q_{t-6} \pmod{127}. \tag{9}$$

This shortcut has active lags $\{2, 6\}$ and a delay of $6 - 3 = 3$.

C.2 Existence of Shortcuts

Periodicity provides a trivial shortcut. Every invertible linear recurrence over a finite field admits a shortcut due to its periodicity. Consider an order-k recurrence

$$q_t = \sum_{i=1}^{k} a_i q_{t-i} \pmod{p}, \tag{10}$$

where $k \geq 2$ and $a_k \neq 0$. Let $s_t = (q_t, \ldots, q_{t+k-1})^\top$ denote its state vector, with state transition $s_{t+1} = As_t$. Since $\det A = (-1)^{k-1} a_k \neq 0$, the transition matrix A is invertible.

As there are finitely many invertible matrices over $\mathbb{F}_p$, there exists a positive integer T such that $A^T = I$. Consequently, every sequence generated by the recurrence satisfies $q_t = q_{t-T}$ whenever both terms are defined. Choosing a positive multiple $M > k$ of T gives

$$q_t = q_{t-M}, \qquad t > M. \tag{11}$$

The existence of general shortcuts is guaranteed by algebraic properties. For an order-k recurrence, the existence of shortcuts can be characterized algebraically using its characteristic polynomial. By representing polynomial remainders as vectors over a finite field, the shortcut condition can be expressed as a linear dependence condition. The pigeonhole principle then guarantees that such a dependence occurs, establishing the existence of shortcuts for arbitrary recurrence orders.

D Identifying Recurrence Formulas Learned by Transformers

D.1 Experimental Setup

To identify the recurrence formulas learned by Transformers, we combine attention analysis with controlled input interventions. This approach allows us to investigate both alternative formulas used under masked observations (Appendix D.2) and shortcuts learned in higher-order tasks (Appendix D.3). Our method consists of two stages: identifying candidate historical positions from attention patterns and recovering the algebraic relationships between these positions and the model's predictions.

The central idea is to construct intervention inputs that do not necessarily satisfy the original recurrence. In valid recurrence sequences, historical values are algebraically dependent, so multiple equivalent formulas may produce identical predictions. Consequently, prediction accuracy on valid sequences alone cannot reveal which historical values the model actually uses. By independently varying the input values, we break these dependencies and examine the model's input-output relationship.

Step 1: Identifying candidate positions through attention analysis. We first visualize the attention matrices across different layers and heads. For each prediction position, we collect the historical positions receiving substantial attention and express them as relative distances from the current position.

We aggregate these attention patterns across heads and layers to identify candidate historical positions. Consecutive prediction positions with similar attention patterns are grouped together for subsequent analysis.

Importantly, attention patterns only indicate where the model retrieves information; they do not establish how the retrieved values contribute to its prediction. We therefore treat the attended positions as candidates rather than direct evidence of a particular recurrence formula.

Step 2: Recovering algebraic relationships through off-manifold probing. To identify the computation performed by the model, we construct artificial input sequences in which the observed values are sampled independently and uniformly from $\mathbb{F}_p$. Unlike the original training sequences, these inputs are not constrained to satisfy any recurrence relation. For masked experiments, we preserve the mask pattern, represent masked positions by $\perp$, and independently sample only the unmasked values.

Independent random inputs remove the algebraic dependencies among historical values, allowing different candidate formulas to be distinguished through their predictions.

For a model predicting q_{t+1}, let D_t denote the set of candidate attention distances, augmented with the three most recent positions used by the original order-3 recurrence. For masked inputs, we retain only the candidate positions that are observable under the given mask pattern.

We examine whether the model's predictions on the artificial inputs satisfy a linear relationship of the form

$$\hat{q}_{t+1} \equiv \sum_{d \in D_t} \beta_d q_{t-d} \pmod{p}, \tag{12}$$

where $\beta_d \in \mathbb{F}_p$ are the coefficients to be recovered.

We collect the model's predictions on independently generated input sequences and fit the coefficients over $\mathbb{F}_p$. We first attempt to solve the resulting system exactly using modular Gaussian elimination. If no exact solution exists, we search for a best-fitting linear formula using RANSAC.

To quantify the quality of a recovered formula, we calculate its prediction agreement:

$$\text{Agreement} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{q} \in \mathcal{D}} \mathbb{I}\left[\hat{q}_{t+1}(\mathbf{q}) = \left(\sum_{d \in D_t} \beta_d q_{t-d}\right) \bmod p\right], \tag{13}$$

where $\mathcal{D}$ is the collection of artificial probe inputs.

The recovered nonzero coefficients identify the historical positions contributing to the fitted formula. We compare the recovered formulas with the original recurrence and examine how the models use different historical positions in the presence or absence of masking.

Because the probe inputs violate the original recurrence constraints, high agreement cannot be attributed solely to algebraic correlations in the training distribution. This provides stronger evidence of the model's computational behavior than attention visualization or prediction accuracy on valid sequences alone.

D.2 Identifying Alternative Formulas Under Masked Single-rule Recurrence

We apply our rule-identification method to the single-rule setting with masked observations to investigate how models handle missing historical values.

Figure 10 presents three representative masking patterns: $(\perp, \perp, q_{t-1})$, $(\perp, q_{t-5}, \perp, \perp, q_{t-2}, \perp)$, and $(\perp, \perp, q_{t-4}, q_{t-3}, \perp, \perp)$, where $\perp$ denotes a masked token and q an observed value. For each pattern, attention analysis identifies candidate historical positions, while off-manifold probing recovers the corresponding prediction formula.

Although the original recurrence uses the two immediately preceding values with coefficients $(1, 1)$, all three recovered formulas involve alternative historical positions and coefficients. In particular, the third pattern masks both immediately preceding values, yet the model successfully predicts using values from more distant positions. The recovered formulas achieve prediction agreement ranging from 90.0% to 98.8% on the probing inputs.

These results provide evidence that Transformers can learn multiple algebraically equivalent recurrence formulas and select alternative formulas based on the available historical values under different masking patterns. This helps explain how increasing model width can improve generalization under masking, as observed in Fig. 3.

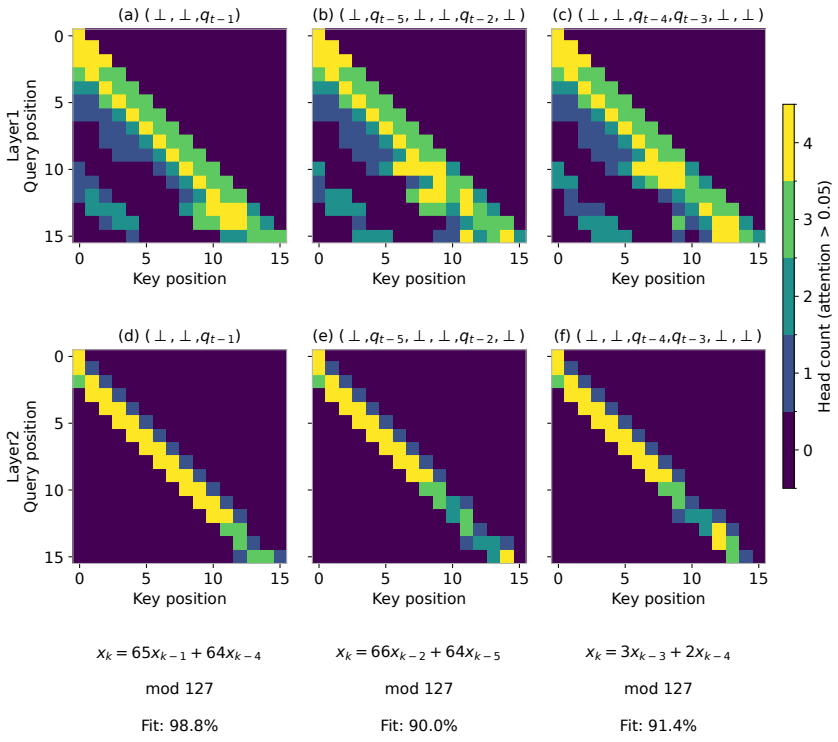


Figure 10: Attention patterns and recovered recurrence formulas under three representative masking patterns in the single-rule setting. Each column corresponds to a different masking pattern, where $\perp$ denotes a masked token and q an observed value. The first two rows show attention maps for the two Transformer layers. Each cell counts the number of heads (out of four) whose attention weight exceeds 0.05. The bottom row reports the recurrence formula recovered through off-manifold probing and its prediction agreement. All three patterns yield alternative formulas distinct from the original $(1, 1)$ recurrence, illustrating how the model exploits different historical positions to predict under missing observations.

D.3 Identifying Shortcuts in Order-3 Tasks

We apply the procedure above to four representative models, selected to illustrate how shortcut delay and loss reweighting affect the recurrence formulas used at different prediction positions.

Small delay (Figure 11(a)). In the first example, the recovered formulas are consistent with the original order-3 recurrence at the earliest prediction positions. Once sufficient historical context becomes available, the recovered formula changes to a two-term shortcut. This example illustrates how a model can use different algebraically equivalent formulas at different positions within the same sequence.

Intermediate delay (Figure 11(c)). We observe a similar transition in the second example, but the region in which the model uses the original recurrence extends over more prediction positions. The recovered formula changes to a shortcut only at later positions, consistent with the longer context required by its shortcut delay.

Large delay (Figure 11(d)). In contrast, the third example shows no transition to a shortcut within the evaluated prediction positions. The recovered formulas remain consistent with the original order-3 recurrence throughout the sequence. This behavior illustrates how a sufficiently long shortcut delay can favor learning the original recurrence within the available context.

Loss reweighting (Figure 11(b)). Finally, we revisit the short-delay example after increasing the loss weight factor of the first prediction position to 40. Unlike the corresponding model trained with the standard loss, the recovered formulas remain consistent with the original recurrence across all

evaluated prediction positions. Thus, although a short shortcut is algebraically available, the training objective can discourage the model from relying on it.

Together, these examples show that shortcut availability alone does not determine which recurrence formula the model learns. The formula used at each prediction position also depends on the training objective.

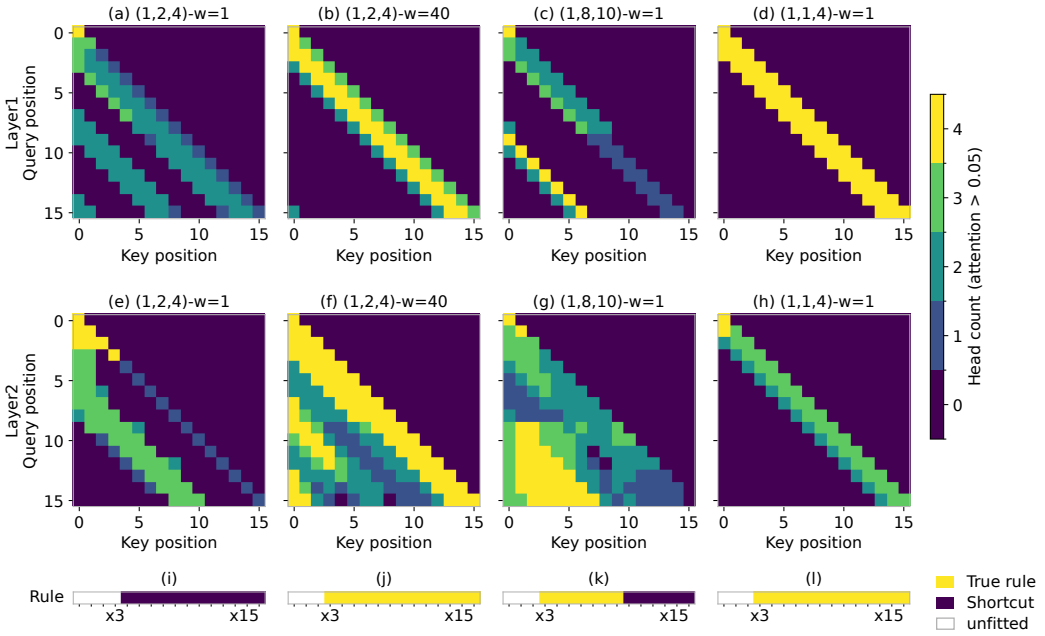


Figure 11: Attention patterns and recovered recurrence rules for four representative order-3 tasks. Each column corresponds to one experiment: shortcut delays of 1, 1, 9, and 13, respectively, with the second experiment using a first-prediction loss weight of $w = 40$ ($w = 1$ otherwise). The first and second rows show attention maps for the two Transformer layers. Each cell counts the number of attention heads (out of four) whose attention weight exceeds 0.05 for the corresponding query and key positions. The bottom row classifies the formula recovered at each prediction position through off-manifold probing as the original recurrence (yellow), a shortcut (purple), or without a reliable fit (white).

E Additional Experiments on Challenging Tasks

This section presents additional experiments on challenging task configurations, including multi-rule learning, masked observations, and higher-order recurrences. We examine the effects of increasing model capacity and investigate how the availability of shortcuts affects learning.

E.1 Four-Rule Inter-trajectory Learning

We investigate whether increasing model depth or width enables successful joint learning of four latent rules. As shown in Figure 12, neither increasing depth to 12 layers nor increasing width to 4096 leads to successful generalization within the tested training budgets. Although the models achieve high training accuracy, their test accuracy remains near chance. These results contrast with curriculum learning, which enables models to acquire additional latent rules by progressively introducing them, as discussed in Section 4.

E.2 Masked Inter-trajectory Learning

We further investigate the combination of latent-rule learning and incomplete observations. Figure 13 shows that this setting requires substantially greater model capacity. Among the four tested

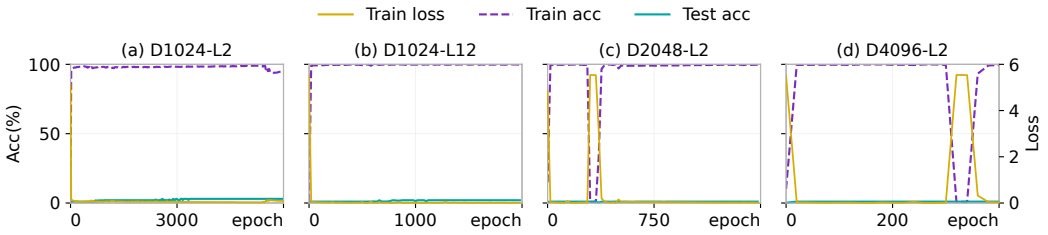


Figure 12: Joint training on four-rule inter-trajectory tasks with different model depths and widths. Panels (a)–(d) correspond to $(d_{\text{model}}, L) = (1024, 2), (1024, 12), (2048, 2), (4096, 2)$, respectively. Each panel reports training loss, training accuracy, and test accuracy. All four configurations achieve high training accuracy but fail to generalize within their respective training budgets.

configurations, only the six-layer model successfully generalizes. Notably, increasing depth from two to six layers enables successful learning, whereas increasing width from 1024 to 4096 does not achieve the same result within the tested training budgets. This suggests that depth and width play different roles in learning masked latent-rule tasks.

We hypothesize that masking further complicates latent-rule identification by removing observations that would otherwise help distinguish different recurrence rules.

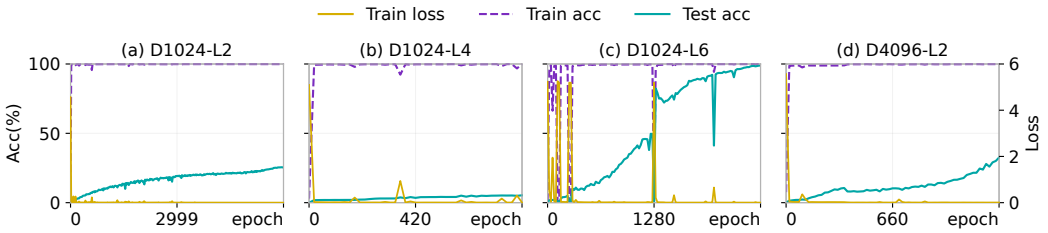


Figure 13: Learning curves for the masked two-rule inter-trajectory setting. Panels (a)–(d) correspond to $(d_{\text{model}}, L) = (1024, 2), (1024, 4), (1024, 6), (4096, 2)$, respectively. Only the six-layer model in panel (c) achieves high test accuracy, while the other configurations fail to generalize within their respective training budgets. Each panel reports training loss, training accuracy, and test accuracy.

E.3 Challenging Order-4 Recurrences

Order-4 recurrences may admit both two-term and three-term shortcuts. Our experiments show that learning order-4 recurrences remains challenging. We observe successful generalization only in cases where models learn a two-term shortcut (Figure 14). However, tasks beyond two-term shortcut are challenging and Transformers fail to generalize within the training budget (Figure 15).

E.4 Order-3 Intra-trajectory Learning

We investigate order-3 recurrences in the intra-trajectory setting, where the active rule may change at every prediction step. Unlike fixed-rule recurrences, these sequences do not generally satisfy the shortcuts derived for a single recurrence rule, since successive values may be generated by different rules.

Consequently, models cannot directly exploit the fixed-rule shortcuts discussed in Section 3.3. Figure 16 shows that all three tested configurations fail to generalize. Increasing either model depth or width does not resolve this difficulty within the evaluated training budgets.

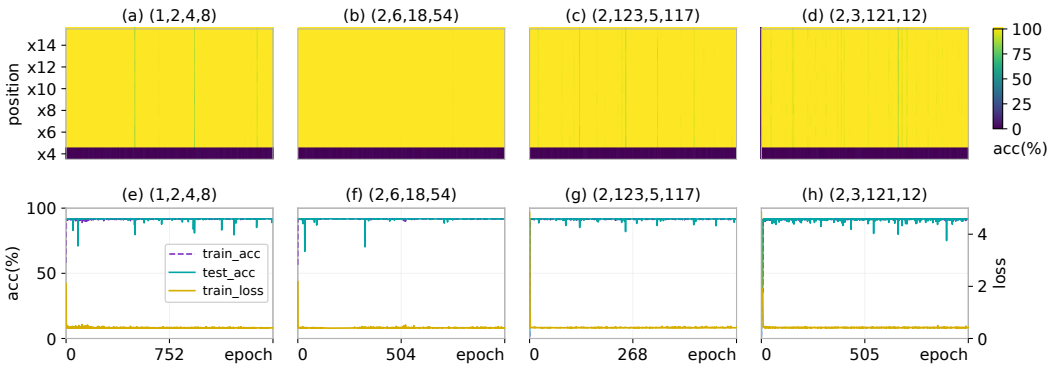


Figure 14: Order-4 recurrences with two-term shortcuts ($p = 127$, $d_{\text{model}} = 1024$, $L = 2$). From left to right, the learned shortcuts are $q_t = 2q_{t-1} - q_{t-5}$, $q_t = 5q_{t-1} + 92q_{t-5}$, $q_t = 124q_{t-3} + 107q_{t-5}$, and $q_t = 7q_{t-2} + 24q_{t-5}$, all modulo 127. The top row shows per-position test accuracy over training epochs, while the bottom row shows the corresponding training accuracy, test accuracy, and training loss.

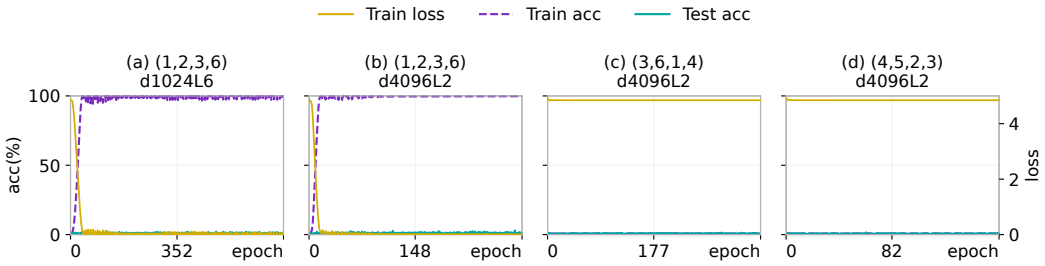


Figure 15: Learning curves for challenging order-4 recurrences over $\mathbb{F}_{127}$. Panels (a) and (b) examine the rule $(1, 2, 3, 6)$ with configurations $(d_{\text{model}}, L) = (1024, 6)$ and $(4096, 2)$, respectively. Panels (c) and (d) examine rules $(3, 6, 1, 4)$ and $(4, 5, 2, 3)$, respectively, using $(d_{\text{model}}, L) = (4096, 2)$. The first two configurations achieve high training accuracy but fail to generalize, while the remaining two fail to fit the training data under their respective training configurations. Curves show training loss, training accuracy, and test accuracy.

E.5 Order-3 Inter-trajectory Learning

Finally, we investigate the interaction between latent-rule identification and higher-order recurrence learning. Figure 17 compares several pairs of order-3 rules with different shortcut properties.

When at least one rule admits a shortcut with delay 1, models can achieve approximately 95% overall test accuracy. The per-position accuracy reveals that these models generalize at later positions but fail at the first prediction position, where the shortcut is not yet applicable. In contrast, when both rules have longer shortcut delays, none of the tested configurations achieves successful generalization.

These observations suggest that short-delay shortcuts can facilitate higher-order multi-rule learning, even when models fail to acquire the original recurrence.

F Extension to Inter-Trajectory Learning with Nonlinear Recurrence Rules

We extend the inter-trajectory setting from linear recurrence rules to mixtures of linear and nonlinear rules. We use the addition rule

$$f_{\text{add}}: \quad q_t = q_{t-1} + q_{t-2} \pmod{127},$$

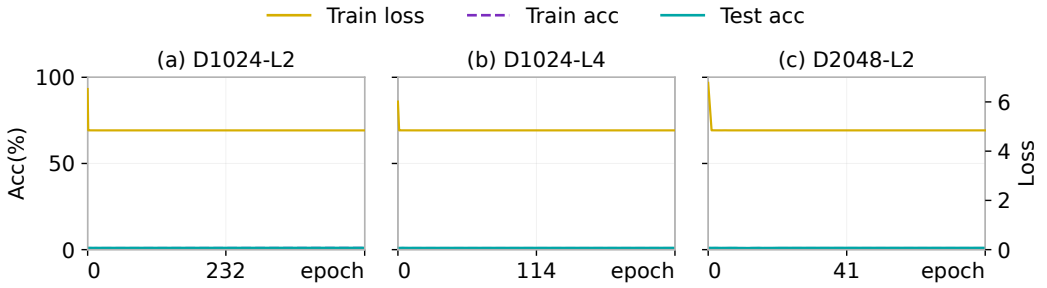


Figure 16: Learning curves for the two-rule, order-3 intra-trajectory setting. Panels (a)–(c) correspond to $(d_{\text{model}}, L) = (1024, 2), (1024, 4), (2048, 2)$, respectively. All three configurations remain near chance test accuracy within their respective training budgets. Curves show training loss, training accuracy, and test accuracy.

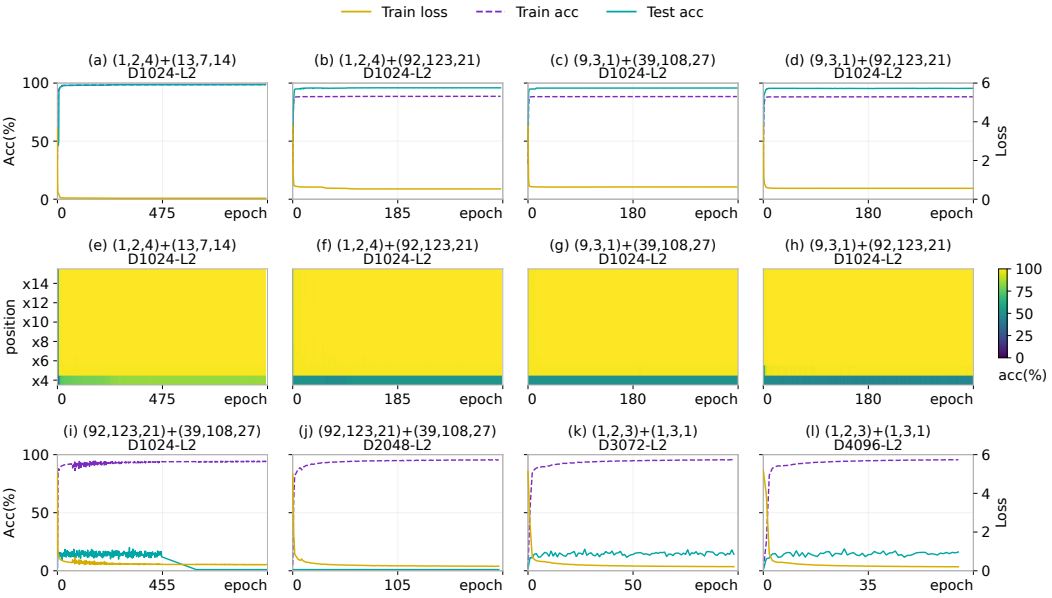


Figure 17: Order-3 inter-trajectory learning with different pairs of recurrence rules over $\mathbb{F}_{127}$. The top row shows learning curves for rule pairs $(1, 2, 4) + (13, 7, 14)$, $(1, 2, 4) + (92, 123, 21)$, $(9, 3, 1) + (39, 108, 27)$, and $(9, 3, 1) + (92, 123, 21)$, respectively, using $d_{\text{model}} = 1024$ and $L = 2$. The middle row shows the corresponding per-position test accuracy. Although these configurations achieve high overall test accuracy by exploiting short-delay shortcuts, they fail at the first prediction position. The bottom row presents additional experiments with longer-delay shortcuts: $(92, 123, 21) + (39, 108, 27)$ with $(d_{\text{model}}, L) = (1024, 2)$ and $(2048, 2)$, and $(1, 2, 3) + (1, 3, 1)$ with $(d_{\text{model}}, L) = (3072, 2)$ and $(4096, 2)$. None of these configurations achieves successful generalization. Learning curves report training loss, training accuracy, and test accuracy.

and consider two multiplication rules:

$$f_{\text{mul}}: \quad q_t = q_{t-1}q_{t-2} \pmod{127}, \qquad f_{\text{mul23}}: \quad q_t = q_{t-1}^2 q_{t-2}^3 \pmod{127}.$$

We study two two-rule settings: $(f_{\text{add}}, f_{\text{mul}})$ and $(f_{\text{add}}, f_{\text{mul23}})$. The second setting follows the same coefficient pattern as the two linear rules $(1, 1)$ and $(2, 3)$ used in our main experiments, allowing us to compare linear and nonlinear rule mixtures under a similar setup. All models are jointly trained from random initialization without curriculum learning, with either 0% or 10% random input masking.

Data, models, and training. We use 70% of the examples from each rule for training and the rest for testing. We evaluate five model configurations: $(d_{\text{model}}, L, H) \in (256, 1, 2), (256, 2, 2), (256, 2, 4), (512, 1, 4), (512, 2, 4)$. We use AdamW with an initial learning rate of 3×10^{-4}, weight decay 1.0, batch size 512, and cosine learning-rate decay over a maximum of 6,000 epochs. Training stops 200 epochs after the overall test accuracy first reaches 99%.

Multiplication is learned faster than addition on $(f_{\text{add}}, f_{\text{mul}})$. Figures 18 and 19 show the learning curves for $(f_{\text{add}}, f_{\text{mul}})$. Across the evaluated model settings and training runs, multiplication reaches high test accuracy earlier than addition, or has higher accuracy when generalization is incomplete. This pattern appears both with and without input masking, showing that the nonlinear multiplication rule is not harder to learn than addition in this setting.

Input masking delays generalization. Generalization occurs later with 10% input masking than without masking, especially for the addition rule. This pattern appears for both nonlinear rule mixtures, as shown by comparing the masked and unmasked results in Figures 18–21. This is consistent with our results for linear rules, where input masking also makes generalization harder.

Nonlinear rules do not necessarily make generalization harder. Figures 20 and 21 show that two-layer models can generalize to both f_{add} and f_{mul23} even with 10% random input masking. In contrast, joint training on the two linear rules $(1, 1)$ and $(2, 3)$ does not reach joint generalization under the same masking rate in our main experiments. Thus, the difficulty of inter-trajectory learning cannot be explained by whether the recurrence rules are linear or nonlinear alone.

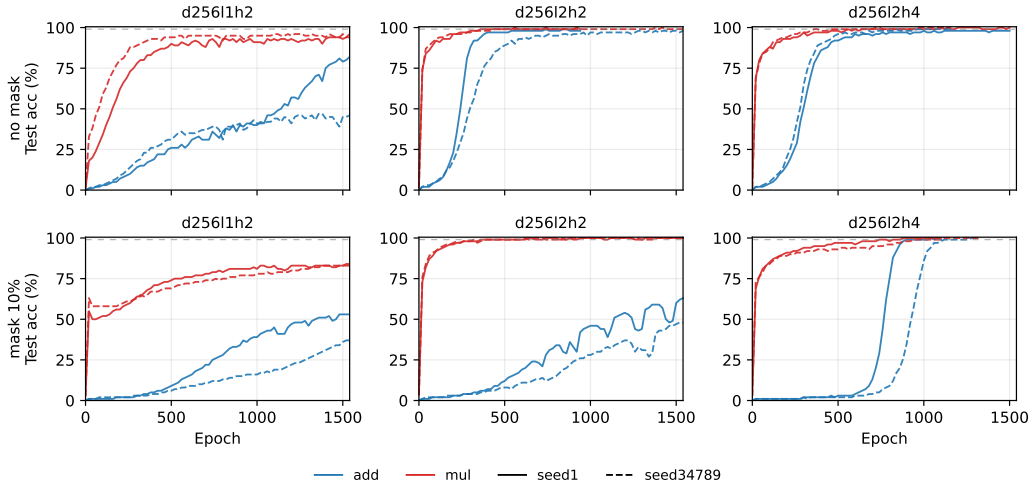


Figure 18: Learning curves for f_{add} and f_{mul} with $d_{\text{model}} = 256$. Subplot titles report d_{model}, L, and H after `d`, `l`, and `h`, respectively.

G Experimental details for how transformers learn, represent, and execute rules

G.1 Experimental Details for Interactions Between Training Fractions

Task and data construction. The experiments in Figure 8 use the inter-trajectory setting with two order-2 rules, with coefficient pairs $(1, 1)/(2, 3)$ and $(2, 7)/(5, 3)$. For each rule, 10% of the data examples form the held-out test block. Each rule's training fraction is selected from $\{0.05, 0.1, 0.3, 0.5, 0.7\}$ and is measured relative to the full p^2 examples per rule. The same held-out test dataset is used across fraction combinations for a given seed.

Model and training. We use the decoder-only Transformer described in the main text, with $d_{\text{model}} = 512$, $L = 2$ layers, and $H = 4$ attention heads. Training uses AdamW with learning rate

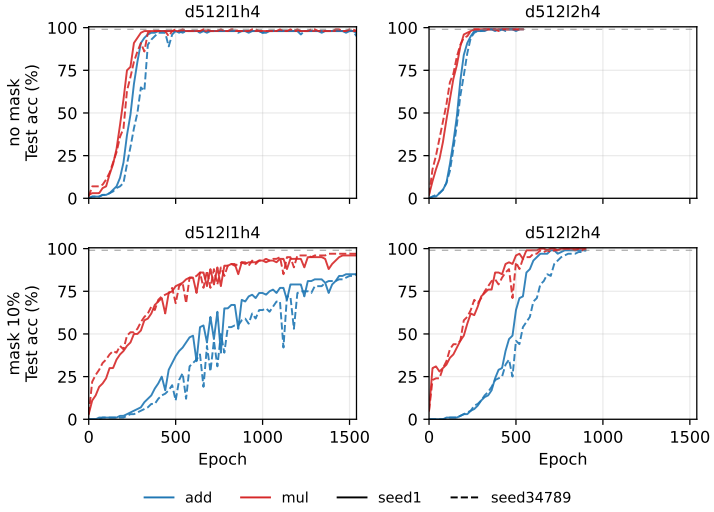


Figure 19: Learning curves for f_{add} and f_{mul} with $d_{\text{model}} = 512$. Subplot titles report d_{model}, L, and H after `d`, `l`, and `h`, respectively.

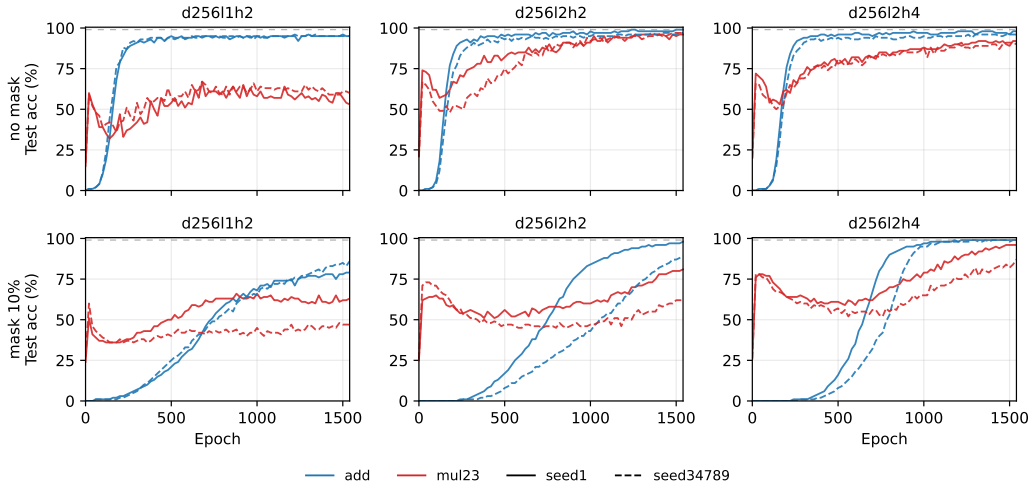


Figure 20: Learning curves for f_{add} and f_{mul23} with $d_{\text{model}} = 256$. Subplot titles report d_{model}, L, and H after `d`, `l`, and `h`, respectively.

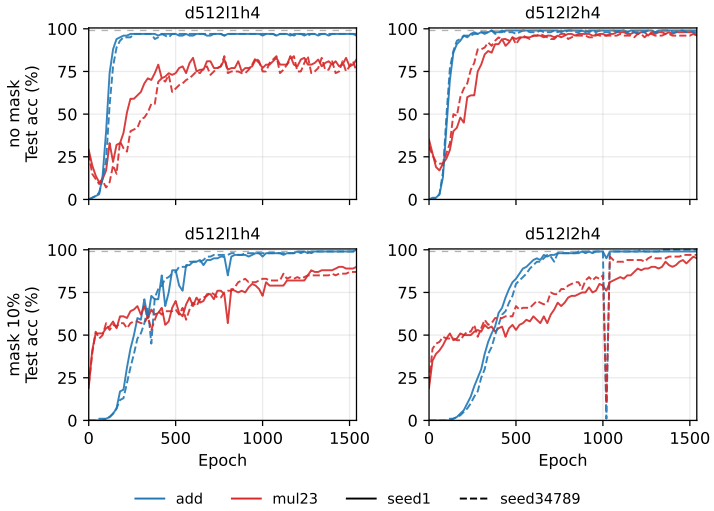


Figure 21: Learning curves for f_{add} and f_{mul23} with $d_{\text{model}} = 512$. Subplot titles report d_{model}, L, and H after `d`, `l`, and `h`, respectively.

3×10^{-4}, weight decay 1.0, batch size 512, and a cosine-annealing schedule over 20,000 epochs. Early stopping occurs only after overall held-out test accuracy reaches 99% for 200 consecutive epochs. Per-rule held-out accuracies are reported at the best epoch of each run.

Random seeds. The left and middle panels of Figure 8 use seed 17996. The right panel reports the mean and standard deviation across seeds $\{17996, 18318, 34789, 44536\}$, with the $(1, 1)$ training fraction fixed at 0.3 and the $(2, 3)$ training fraction varied. The rule labels 0 and 1 in the plots correspond to the first and second rules, respectively, whose training fractions are denoted by r_1 and r_2 in the text.

G.2 Linear probes of rule identity

Task and checkpoints. The probes in Table 1 distinguish the two rules in the unmasked inter-trajectory setting. We evaluate six trained models with $(d_{\text{model}}, L, H) \in (256, 1, 2), (512, 1, 4), (256, 2, 2), (256, 2, 4), (512, 2, 2), (512, 2, 4)$ and sequence length 16. Each model is jointly trained on the two rules using 70% of the data from each rule. We use the final checkpoint of each run and keep the model frozen during probe training.

Probe data and split. For each rule, we generate length-16 sequence windows from all 127^2 initial states, shuffle them with seed 0, and retain the first 2,000. Within each rule, a further random permutation with seed 0 assigns 1,000 sequences to probe training and 1,000 to probe testing. The two rules are permuted separately. Both splits are therefore balanced, containing 2,000 sequences each; binary labels 0 and 1 encode rules $(1, 1)$ and $(2, 3)$, respectively. The same split is used across layers and token positions. The held-out designation refers to probe fitting: these examples are drawn from the full state space, without requiring exclusion from the Transformer's original training set or enforcing disjoint recurrence trajectories between the probe splits.

Activation sites and positions. At each layer, the attention feature concatenates all H head outputs after attention-weighted value aggregation, immediately before the attention output projection. Its dimension is d_{model}. The MLP feature is the hidden vector after the first linear map and GELU, immediately before the second linear map, with dimension rd_{model}. Both features are computed downstream of the respective sublayer's input LayerNorm and upstream of its output projection and residual addition. Neither is a normalized residual-stream output. In particular, the MLP feature is not the final MLP output. We extract activations in evaluation mode and fit a separate probe for every checkpoint, layer, site, and input position. Each probe receives one vector per sequence, with

no concatenation or averaging across positions. We number layers from 1 in the paper. The stored activation keys number them from 0.

Probe fitting and evaluation. We standardize each feature coordinate using the probe training set only and apply the fitted transformation to the test set. A logistic-regression classifier with an intercept and ℓ_2 regularization is then fitted using `scikit-learn`, with inverse regularization strength $C = 1$, solver `lbfgs`, and at most 1,000 iterations. AUROC uses the predicted probability of rule $(2, 3)$. Classification accuracy uses a probability threshold of 0.5. Chance performance is 0.5 for both metrics on these balanced splits. The retained results use one probe split per checkpoint, with no repeated-seed uncertainty estimate. The separate five-fold cross-validation notebook in the code archive uses a different sampling protocol and is not the source of the reported numbers.

Aggregation and layer dependence. The position-specific results cover all 16 input tokens. Tables 1 average metrics over $\{q_3, \cdots, q_{16}\}$, weighting each position equally. This excludes the two initial values, before any recurrence transition has been observed, and includes every subsequent position rather than selecting the best-performing position. Metrics are computed separately at each position before averaging. Predictions from different probes are not pooled to compute a single AUROC. These positional averages are not averages over independent training runs.

Table 1 reports both metrics at every layer. First-layer AUROC remains near 0.5 in all six configurations, whereas second-layer MLP AUROC is high in every two-layer model. Attention decodability is more position-dependent: for $(d_{\text{model}}, L, H) = (256, 2, 4)$, second-layer attention AUROC increases from 0.789 at the third token to 0.933 at the fourth and at least 0.974 thereafter; for $(512, 2, 2)$, it rises from 0.657 at the third token to 0.993 at the sixteenth. Thus, averaging across positions includes the early prefixes where rule information is less readily decoded.

Table 1: Rule-probe results at every layer, averaged over the $\{q_3, \cdots, q_{16}\}$ input tokens. Attention and MLP hidden features use the sites specified above. Accuracies are fractions.

(d_{model}, L, H)	Layer	AUROC		Accuracy	
		Attention	MLP hidden	Attention	MLP hidden
$(256, 1, 2)$	1	0.492	0.480	0.494	0.487
$(512, 1, 4)$	1	0.487	0.487	0.490	0.493
$(256, 2, 2)$	1	0.491	0.478	0.492	0.484
	2	0.502	0.998	0.502	0.984
$(256, 2, 4)$	1	0.489	0.489	0.492	0.492
	2	0.967	0.999	0.916	0.990
$(512, 2, 2)$	1	0.491	0.489	0.495	0.493
	2	0.942	0.997	0.890	0.978
$(512, 2, 4)$	1	0.492	0.491	0.495	0.494
	2	0.991	0.996	0.978	0.985

G.3 Local Recurrence Tests

We test whether the models use the intended local recurrence by breaking the earlier trajectory while keeping the information needed for local prediction. Figure 22 illustrates the test. The construction differs slightly between the inter- and intra-trajectory settings because the rule must be inferred from values in the former but is directly given by the rule tag in the latter.

G.3.1 Two-rule inter-trajectory setting

We construct inputs with a random early prefix followed by a short rule-consistent suffix. Using sampling seed 42, we sample 200 prefixes with $q_1, \ldots, q_6$ independently and uniformly drawn from $\mathbb{F}_{127}$. For each rule f_a, we then generate

$$q_7 = f_a(q_5, q_6), \qquad q_8 = f_a(q_6, q_7),$$

and test whether the model predicts

$$q_9 = f_a(q_7, q_8).$$

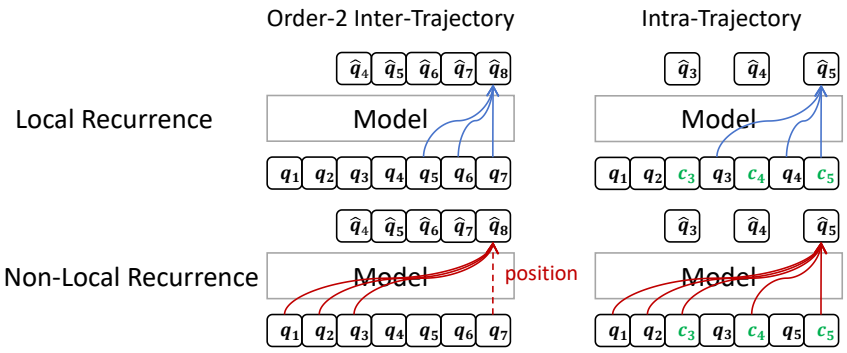


Figure 22: Illustration of the local recurrence test. We break the earlier trajectory while keeping the recent values needed for local prediction. In the inter-trajectory setting, the recent values must also identify the rule. In the intra-trajectory setting, the rule is given by the current rule tag.

The model receives $(q_1, \ldots, q_8)$ without a rule tag. Thus, the full trajectory is generally invalid, while the recent values (q_5, q_6, q_7, q_8) follow the selected rule. Table 2 reports the prediction accuracy. We use the same random prefixes for all rules and checkpoints.

Table 2: Accuracy on the local recurrence test in the two-rule inter-trajectory setting. Each model is evaluated on 200 random prefixes per rule.

d_{model}	L	H	Rule $(1,1)$	Rule $(2,3)$
256	2	2	0.995	0.995
256	2	4	0.995	0.995
512	1	4	0.995	1.000
512	2	2	0.995	1.000
512	2	4	0.980	0.985

We further test whether the prediction depends on the value before the recent rule-consistent triple. We vary q_5 over all 127 values while keeping all other input values, including (q_6, q_7, q_8), fixed. This breaks the transition into q_7 for most values of q_5 while preserving the final triple (q_6, q_7, q_8) and the target $q_9 = f_a(q_7, q_8)$. Across the 25,400 inputs per rule and checkpoint, accuracy ranges from 91.6% to 99.6%; all models except the $d_{\text{model}} = 512$, $L = 2$, $H = 4$ model remain above 95%. Thus, the recent valid triple is enough for high prediction accuracy, although predictions are not fully independent of the earlier context.

G.3.2 Two-rule intra-trajectory setting

We test local recurrence more directly in the intra-trajectory setting by randomizing all value tokens. The input keeps the original form

$$[q_1, q_2, c_3, q_3, c_4, q_4, \ldots],$$

where each q_t is sampled independently and uniformly from $\mathbb{F}_{127}$ and each rule tag c_t is sampled as in training. Let σ_t denote the rule given by c_t. No value is masked during evaluation, including for models trained with input masking.

At each rule tag c_t, we compare the model prediction $\hat{q}_t$ with the local target

$$q_t^{\text{local}} = f_{\hat{c}_t}(q_{t-2}, q_{t-1}).$$

Because the value tokens are sampled independently, this target generally differs from the random token q_t that follows c_t. For B test sequences, we measure

$$\text{Acc}_{\text{local}}(t) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\left[\hat{q}_t^{(b)} = f_{\hat{c}_t^{(b)}}\left(q_{t-2}^{(b)}, q_{t-1}^{(b)}\right)\right].$$

Predictions are evaluated from the supplied prefix and are not fed back into later positions.

Table 3: Agreement with the local recurrence on randomized-value inputs in the two-rule intra-trajectory setting. Each row represents one training run.

Mask rate	Run	All rules	Rule $(1, 1)$	Rule $(2, 3)$
0%	A	1.00	1.00	1.00
0%	B	1.00	1.00	1.00
0%	C	1.00	1.00	1.00
0%	D	1.00	1.00	1.00
10%	A	0.64	0.29	1.00
10%	B	1.00	1.00	1.00
10%	C	0.80	0.59	1.00
10%	D	1.00	1.00	1.00

G.3.3 Visualization of Attention Scores

We visualize the attention weights after softmax for the inter- and intra-trajectory settings. Figures 23–25 show the attention patterns for the unmasked inter-trajectory, unmasked intra-trajectory, and masked intra-trajectory settings, respectively. In all three settings, attention is not limited to the recent values needed by the local recurrence; earlier tokens can also receive nonzero attention. However, nonzero attention does not imply that a token is needed for prediction. We therefore use these figures only to show the attention patterns and rely on the intervention experiments in Section 5.3 to test the local recurrence hypothesis.

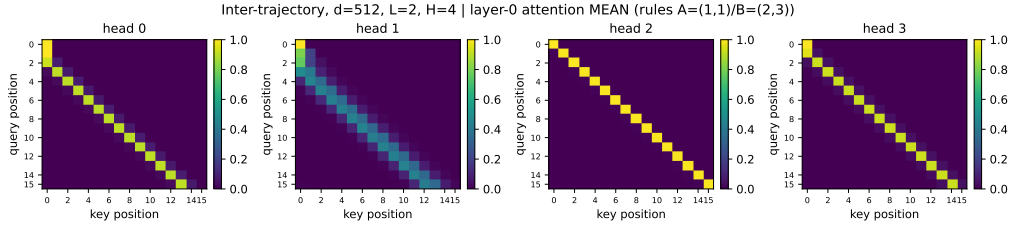


Figure 23: Attention scores in the unmasked two-rule inter-trajectory setting. Earlier values also receive nonzero attention; for example, Head 1 attends to q_{t-4} when predicting q_t at position q_{t-1}.

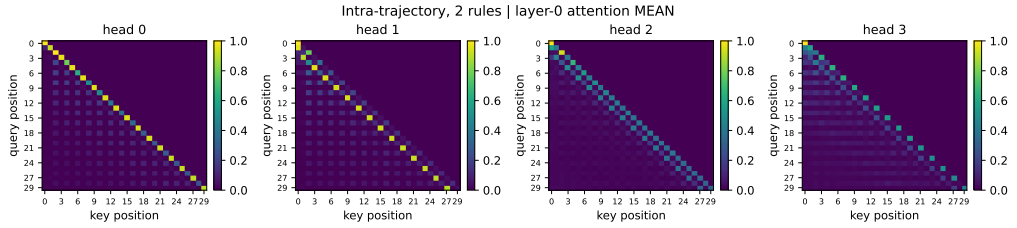


Figure 24: Attention scores in the unmasked two-rule intra-trajectory setting. The model is trained without input masking.

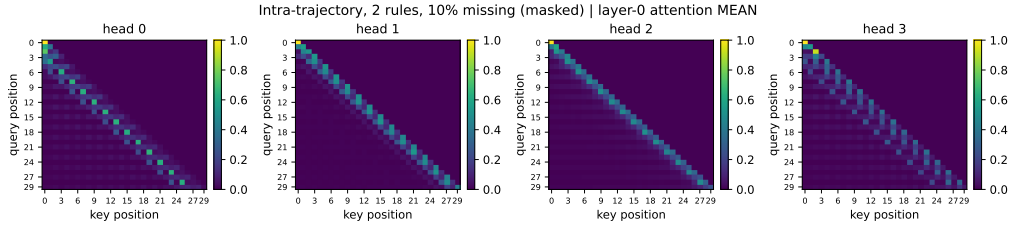


Figure 25: Attention scores in the two-rule intra-trajectory setting with 10% random input masking during training.

H EXPRESSIVITY OF TWO-LAYER CAUSAL TRANSFORMERS

We give an explicit construction for the unmasked inter-trajectory setting of Section 3. A two-layer causal Transformer can predict the next value for every initial state and every rule in a fixed pool, once it has observed $2k$ values. The construction first checks each candidate rule at individual positions and then combines the most recent checks. We specify the architecture, count its resources, and prove correctness for ordinary softmax attention with finite parameters.

H.1 TASK AND PRECISE CAUSAL TRANSFORMER MODEL

Task and quantifiers. We retain the notations from Section 2, and write the fixed pool of K rules as

$$\mathcal{A} = \{a^{(1)}, \dots, a^{(K)}\} \subseteq \mathbb{F}_p^k, \qquad a^{(z)} = (a_1^{(z)}, \dots, a_k^{(z)}).$$

Here $p \geq 2$ is prime, $k \geq 1$ is the order of the recurrence, and $1 \leq K \leq p^k$ is the number of rules. We suppress the trajectory index j. A legal trajectory is generated by one fixed, unobserved $z \in \{1, \dots, K\}$ and arbitrary initial values $q_1, \dots, q_k \in \mathbb{F}_p$, according to

$$q_t = \sum_{i=1}^{k} a_i^{(z)} q_{t-i} \pmod p, \qquad t \geq k+1. \tag{14}$$

Field elements and coefficients are represented by integers in $\{0, \dots, p-1\}$. All network computations take place over the real numbers. We also denote N as the length of the q sequence.

Architecture and dimensions. We use L to denote the layer of the Transformer, each containing causal multi-head self-attention and a position-wise feed-forward network, with residual connections around both sublayers. For brevity, write $d = d_{\text{model}}$ and $d_h = d/H$. We require d_h to be an even integer. The network uses ReLU activation, omits normalization in the blocks and at the output. We use ReLU in the construction in order to simplify the construction. Our results also hold for GELU functions used in our experiments, see appendix H.8 for details.

Each value token has a learned embedding in $\mathbb{R}^d$. We choose the embedding table to realize

$$h_t^{(0)} = (1, q_t, 0, \dots, 0)^{\mathsf{T}} \in \mathbb{R}^d. \tag{15}$$

The first coordinate is a shared constant, independent of position. There are no additive positional features. In block ℓ, head h has matrices $W_{Q,h}^{(\ell)}, W_{K,h}^{(\ell)}, W_{V,h}^{(\ell)} \in \mathbb{R}^{d_h \times d}$. Define

$$\mathbf{Q}_{t,h}^{(\ell)} = W_{Q,h}^{(\ell)} h_t^{(\ell-1)}, \qquad \mathbf{K}_{t,h}^{(\ell)} = W_{K,h}^{(\ell)} h_t^{(\ell-1)}, \qquad \mathbf{V}_{t,h}^{(\ell)} = W_{V,h}^{(\ell)} h_t^{(\ell-1)}. \tag{16}$$

The bold query and key vectors are distinct from the scalar trajectory values q_t and pool size K.

RoPE and the causal mask. Let

$$\begin{aligned} \mathrm{Rot}(u) &= \begin{pmatrix} \cos u & -\sin u \\ \sin u & \cos u \end{pmatrix}, \\ R_t &= \mathrm{diag}\big(\mathrm{Rot}(t\omega_0), \dots, \mathrm{Rot}(t\omega_{d_h/2-1})\big), \qquad \omega_b = 10000^{-2b/d_h}. \end{aligned} \tag{17}$$

In particular, $\omega_0 = 1$. For $1 \leq s \leq t$, the attention scores, weights, and head outputs are

$$A_{t,s,h}^{(\ell)} = \frac{(R_t \mathbf{Q}_{t,h}^{(\ell)})^{\mathsf{T}} (R_s \mathbf{K}_{s,h}^{(\ell)})}{\sqrt{d_h}}, \tag{18}$$

$$\alpha_{t,s,h}^{(\ell)} = \frac{\exp A_{t,s,h}^{(\ell)}}{\sum_{u=1}^{t} \exp A_{t,u,h}^{(\ell)}}, \qquad o_{t,h}^{(\ell)} = \sum_{s=1}^{t} \alpha_{t,s,h}^{(\ell)} \mathbf{V}_{s,h}^{(\ell)}. \tag{19}$$

Note that positions $s > t$ are excluded by the causal mask.

Define $\mathrm{ReLU}(u) = \max\{u, 0\}$ acting coordinatewise, an attention layer is

$$\widetilde{h}_t^{(\ell)} = h_t^{(\ell-1)} + W_O^{(\ell)} \,\mathrm{concat}_{h=1}^{H} o_{t,h}^{(\ell)}, \tag{20}$$

$$h_t^{(\ell)} = \widetilde{h}_t^{(\ell)} + W_2^{(\ell)} \,\mathrm{ReLU}\big(W_1^{(\ell)} \widetilde{h}_t^{(\ell)} + b_1^{(\ell)}\big) + b_2^{(\ell)}. \tag{21}$$

Here $W_O^{(\ell)} \in \mathbb{R}^{d\times d}$, $W_1^{(\ell)} \in \mathbb{R}^{w_\ell \times d}$, $W_2^{(\ell)} \in \mathbb{R}^{d\times w_\ell}$, $b_1^{(\ell)} \in \mathbb{R}^{w_\ell}$, and $b_2^{(\ell)} \in \mathbb{R}^d$. Thus w_ℓ is the hidden width of a single-hidden-layer feed-forward network (FFN), and $d_{\text{ff}} = \max_\ell w_\ell$. All maps are shared across positions; weights may differ across blocks and heads. The final logits and prediction are

$$\lambda_v(q_{1:n}) = (W_{\text{out}} h_n^{(L)})_v, \quad v \in \{0,\dots,p-1\}, \qquad \widehat{q}_{n+1} = \operatorname*{arg\,max}_{v\in\{0,\dots,p-1\}} \lambda_v(q_{1:n}), \tag{22}$$

where $W_{\text{out}} \in \mathbb{R}^{p\times d}$.

H.2 Expressiveness of the Transformer

Define the constants

$$B = p-1, \qquad U = kB^2, \qquad J = \lfloor U/p \rfloor. \tag{23}$$

Theorem H.1 (Two-layer construction) *For every fixed $(p,k,\mathcal{A})$ and every $N \geq 2k$, there is a Transformer of the class in Section H.1 with*

$$L = 2, \qquad H = k, \qquad d_{\text{model}} = 2k\left\lceil \frac{kK+k+4}{2k} \right\rceil, \qquad d_{\text{ff}} \leq 3K(J+1), \tag{24}$$

such that, simultaneously for every legal prefix $q_{1:n}$ with $2k \leq n \leq N$,

$$\lambda_{q_{n+1}}(q_{1:n}) - \lambda_v(q_{1:n}) \geq \tfrac{1}{2} \qquad \text{for every } v \neq q_{n+1}. \tag{25}$$

All parameters are finite real numbers and independent of the active rule, initial state, and actual prefix length. Consequently, the network predicts the next token exactly, with residual width $O(kK+k)$, per-head dimension $O(K+1)$, and FFN hidden width $O(Kkp)$.

In the next sections, we provide the construction and prove the theorem.

H.3 Continuation without unique rule identification

Lemma H.2 (Unique continuation from $2k$ values) *If two infinite sequences satisfy respective order-at-most-k linear recurrences over C_p and agree on their first $2k$ values, then they agree at every subsequent position.*

Proof. Let T be the forward shift, $(Tq)_t = q_{t+1}$. The monic polynomial

$$P_a(X) = X^k - \sum_{i=1}^{k} a_i X^{k-i}$$

annihilates a trajectory with rule a: $P_a(T)q = 0$ at every index $t \geq 1$. For trajectories q, q' with respective rules a, a', the polynomials in T commute, so

$$P_a(T)P_{a'}(T)(q-q') = 0.$$

Their product is monic of degree $2k$. Hence every difference at position $t+2k$ is a linear combination of the preceding $2k$ differences. Since the first $2k$ differences vanish, induction gives $q - q' = 0$ everywhere. No nonzero constant coefficient or recurrence invertibility is needed. ∎

Corollary H.3 (Screening on the most recent window) *For a legal prefix of length $n \geq 2k$, let*

$$\mathcal{C}_n = \left\{ z \in \{1,\dots,K\} : q_t = \sum_{i=1}^{k} a_i^{(z)} q_{t-i} \pmod p \text{ for } t = n-k+1,\dots,n \right\}. \tag{26}$$

Then $\mathcal{C}_n$ is nonempty, and every $z \in \mathcal{C}_n$ yields the same correct next value q_{n+1}.

Proof. The generating rule belongs to $\mathcal{C}_n$. Initialize any candidate $z \in \mathcal{C}_n$ with the k values $q_{n-2k+1},\dots,q_{n-k}$. Its next k generated values agree with $q_{n-k+1},\dots,q_n$ by the defining checks. The candidate trajectory and the true trajectory starting at position $n-2k+1$ therefore agree for their first $2k$ values. Lemma H.2 implies that their next values agree. ∎

The rule itself need not be identifiable: the all-zero trajectory is compatible with every rule. The construction retains all compatible rules and uses their common continuation. This distinction is necessary for a guarantee uniform over initial states.

H.4 EXPLICIT FEED-FORWARD CIRCUITS

We use three scalar piecewise-linear functions:

$$\tau(u) = \mathrm{ReLU}(u+1) - 2\,\mathrm{ReLU}(u) + \mathrm{ReLU}(u-1), \tag{27}$$

$$g(u) = 1 - \sum_{b=0}^{J} \tau(u - bp), \tag{28}$$

$$F(u) = \mathrm{ReLU}(u) - p \sum_{b=1}^{J} \big[\mathrm{ReLU}(u - bp + 1) - \mathrm{ReLU}(u - bp)\big]. \tag{29}$$

Empty sums are zero, including when $J = 0$.

Lemma H.4 (Modular arithmetic and stability) *For every integer $u \in [-B, U]$, $g(u) = \mathbf{1}[u \not\equiv 0 \pmod p]$. On the entire real line, $0 \le g \le 1$ and $\mathrm{Lip}(g) \le 1$. For every integer $u \in [0, U]$, $F(u) = u \bmod p$, with the remainder in $\{0, \dots, p-1\}$. Furthermore, $F(u) = 0$ for $u \le 0$ and $\mathrm{Lip}(F) \le B$ globally. The functions g and F require $3(J+1)$ and $1 + 2J$ hidden ReLU units, respectively.*

Proof. The function $\tau(u) = \max\{1 - |u|, 0\}$ equals one at the integer zero and zero at every other integer. The multiples of p in $[-B, U]$ are precisely $0, p, \dots, Jp$, which proves the integer formula for g. Since $p \ge 2$, the positive supports of the tents in (28) are disjoint. Their sum lies in $[0, 1]$ and has slope in $\{-1, 0, 1\}$ on each linear piece. This proves the global range and Lipschitz bounds for g.

For integer $u \ge 0$,

$$\mathrm{ReLU}(u - bp + 1) - \mathrm{ReLU}(u - bp) = \mathbf{1}[u \ge bp].$$

For $u \le U$, summing over $1 \le b \le J$ gives $F(u) = u - p\lfloor u/p \rfloor$. All terms vanish when $u \le 0$. The descending intervals $(bp - 1, bp)$ are pairwise disjoint, so the only slopes of F are 0, 1, and $1 - p$. Their absolute values are at most B. The displayed formulas give the stated unit counts directly, with an output bias for the constant in g. ∎

Lemma H.5 (Candidate selection and linear readout) *Suppose $E_z \ge 0$ and $u_z \in [0, U]$ are integers for $z = 1, \dots, K$, at least one E_z is zero, and all candidates with $E_z = 0$ satisfy $u_z \bmod p = y$. Set*

$$\xi_z = u_z - (U+1)E_z, \qquad S = \sum_{z=1}^{K} F(\xi_z), \qquad C = \sum_{z=1}^{K} \mathrm{ReLU}(1 - E_z), \tag{30}$$

$$\lambda_v = 2vS - v^2 C, \qquad v = 0, \dots, p-1. \tag{31}$$

Then $S = Cy$, $C \ge 1$, and $\lambda_y - \lambda_v = C(y - v)^2 \ge 1$ for every $v \ne y$. If the u_z, E_z are affine functions of the FFN input, a single-hidden-layer FFN of width $2K(J+1)$ computes (S, C).

Proof. When $E_z = 0$, the contributions to (S, C) are $(y, 1)$. When $E_z \ge 1$, $\xi_z \le U - (U+1) = -1$, and both contributions vanish. Thus $S = Cy$ with $C \ge 1$. Substitution into (31) gives the claimed logit difference. For each candidate, $F(\xi_z)$ uses $1 + 2J$ ReLU units and $\mathrm{ReLU}(1 - E_z)$ uses one more. All preactivations are affine in the FFN input, so no additional hidden layer is needed. The FFN output map forms the two sums directly. ∎

H.5 ROPE HEADS FOR FIXED RELATIVE OFFSETS

Lemma H.6 (Finite-softmax offset retrieval) *For a fixed maximum context length N, define*

$$\Delta_N = \min_{\substack{u \in \mathbb{Z} \\ 1 \le |u| \le N}} (1 - \cos u) > 0, \qquad \eta_\beta = N \exp(-\beta \Delta_N). \tag{32}$$

For any offset $i \in \{1, \dots, k\}$ and $\beta > 0$, a head can have content-independent scores

$$A_{t,s} = \beta \cos(t - s - i). \tag{33}$$

If $t - i \geq 1$ and $t \leq N$, its attention mass at $t - i$ is at least $1 - \eta_\beta$. If each value coordinate lies in an interval of length D, then

$$\|o_t - \mathbf{V}_{t-i}\|_\infty \leq D\eta_\beta. \tag{34}$$

The comparison is to the actual value vector supplied to the head at position $t - i$.

Proof. Using only the first two RoPE coordinates, set

$$c_\beta = (\beta\sqrt{d_h})^{1/2}, \qquad \mathbf{Q} = c_\beta \operatorname{Rot}(-i)(1,0)^\mathsf{T}, \qquad \mathbf{K} = c_\beta(1,0)^\mathsf{T},$$

with zeros in all remaining coordinates. The shared constant in (15) generates these vectors through linear query and key maps. It will be preserved in every residual update. Since the first RoPE frequency is one, their scaled inner product is (33). The target has score β. Every other position has a nonzero integer displacement from the target of absolute value at most N. No nonzero integer is a multiple of 2π, so the finite minimum Δ_N is positive and every other score is at most $\beta(1 - \Delta_N)$. Therefore

$$1 - \alpha_{t,t-i} \leq \sum_{s \neq t-i} \exp(A_{t,s} - \beta) \leq N \exp(-\beta\Delta_N) = \eta_\beta.$$

Each coordinate of $o_t - \mathbf{V}_{t-i}$ is a weighted sum of differences of magnitude at most D, with total weight at most η_β. This proves (34). ∎

If $t - i < 1$, the head still returns a well-defined convex combination of the available values. We will use the retrieval guarantee only when the target exists, and the convex-hull bound at all other positions.

H.6 Construction of the two layers

All active heads use the score in (33), with the same scale β. We first analyze an auxiliary computation in which a head copies its target exactly whenever that target exists. This is only an intermediate step in the proof. The actual network uses finite-softmax attention, whose errors are bounded in Section H.7.

Layer 1: local recurrence checks. Head i, for $i = 1, \ldots, k$, selects the original scalar-value coordinate at offset i. The attention output map writes these values into k separate unused residual coordinates. Denote them by $w_{t,1}, \ldots, w_{t,k}$, and let $w_{t,0} = q_t$, which is preserved by the residual connection. At $t \geq k + 1$, exact retrieval gives $w_{t,i} = q_{t-i}$ for all $0 \leq i \leq k$. The first FFN writes

$$e_{t,z} = g\left(\sum_{i=1}^{k} a_i^{(z)} w_{t,i} - q_t\right), \qquad z = 1, \ldots, K, \tag{35}$$

into K fresh coordinates. For $t \geq k + 1$, these are the binary indicators that rule z violates the observed transition. The affine argument lies in $[-B, U]$, and Lemma H.4 applies. Its implementation uses $3K(J + 1)$ hidden units; each coefficient $a_i^{(z)}$ is a weight in the first affine map, and each integer offset bp is a bias.

At earlier positions, some retrieved values do not correspond to the requested lags. Equation (35) is nevertheless applied unchanged. These early outputs need not represent valid checks. At every position, and also for the actual softmax network, $w_{t,i} \in [0, B]$ and $e_{t,z} \in [0, 1]$.

Layer 2: screening and next-value computation. For $j = 1, \ldots, k - 1$, head j selects the entire K-dimensional indicator vector at offset j and writes it to its own fresh group of residual coordinates. The remaining head has zero value projection. The residual connection retains the current indicator vector and the values $w_{t,0}, \ldots, w_{t,k}$. At the final position n, the second FFN forms, directly in its affine preactivations,

$$E_z = \sum_{j=0}^{k-1} e_{n-j,z}, \qquad u_z = \sum_{i=1}^{k} a_i^{(z)} w_{n,i-1}. \tag{36}$$

The indicator vectors with $j \geq 1$ in this formula denote the retrieved vectors. In the auxiliary exact computation,

$$n - j \geq n - k + 1 \geq k + 1 \qquad (0 \leq j \leq k - 1),$$

so every check being used is valid and $E_z = 0$ precisely when $z \in \mathcal{C}_n$. Also $u_z = \sum_{i=1}^{k} a_i^{(z)} q_{n+1-i} \in [0, U]$. Corollary H.3 gives a nonempty set of surviving candidates, all with $u_z \bmod p = q_{n+1}$.

The FFN implements (30) and writes only the two statistics S, C into fresh coordinates. In particular, E_z, u_z, and ξ_z need no separate residual coordinates or extra layers. Lemma H.5 gives hidden width $2K(J+1)$. The linear output head has entries $2v$ and $-v^2$ on the S and C coordinates, respectively, and zero elsewhere. The auxiliary exact computation therefore has correct-class margin at least one.

Residual width and value dimensions. The construction uses the following disjoint residual coordinates:

Stored quantity	Number of coordinates
Shared constant	1
Original value and k retrieved values	$k+1$
Current indicators and $k-1$ retrieved indicator vectors	kK
Output statistics S, C	2
Total	$kK + k + 4$

Each sublayer writes only into its reserved, initially zero coordinates. In particular, it preserves the constant and original value. Rounding the total up to a multiple of $2k$ gives precisely the value of d_{model} in (24). With $H = k$, the head dimension $d_h = d_{\text{model}}/k$ is even and at least $\max\{2, K\}$. Thus the first layer can use a two-coordinate RoPE pair, and each active second-layer head can transport all K indicators within its value dimension. This value dimension is included in the width bound. The FFN widths are $3K(J+1)$ and $2K(J+1)$; a common FFN width can be obtained by adding unused units. When $k = 1$, the second layer has no active attention head; the same FFN and residual construction still applies.

H.7 Correctness for finite-softmax attention

We now prove that the actual network, with the explicit ReLU functions above and finite attention parameters, preserves the classification margin. A superscript $\star$ denotes an ideal quantity from the auxiliary exact computation at positions where its lagged inputs exist; a hat denotes the corresponding actual quantity. Set

$$D_E = k(U+1), \qquad D_\xi = U + (U+1)D_E, \qquad M = KB^2(2D_\xi + D_E). \tag{37}$$

First-layer error. At each $t \geq k+1$, Lemma H.6 gives $|\widehat{w}_{t,i} - q_{t-i}| \leq B\eta_\beta$. The original q_t coordinate is exact. Hence the error in the affine argument of g in (35) is at most

$$\sum_{i=1}^{k} a_i^{(z)} B\eta_\beta \leq kB^2\eta_\beta = U\eta_\beta.$$

Since g is 1-Lipschitz, $|\widehat{e}_{t,z} - e^\star_{t,z}| \leq U\eta_\beta$ at every such position. At every position, including the early positions, $\widehat{e}_{t,z} \in [0,1]$.

Second-layer error. Each target position $n-j$, for $1 \leq j \leq k-1$, is at least $k+1$. The error of retrieving its actual indicator vector is at most η_β, because all source indicator coordinates lie in $[0,1]$. Its actual vector is itself within $U\eta_\beta$ of the ideal vector. Including the current indicator, which does not require another attention operation, gives

$$|\widehat{E}_z - E^\star_z| \leq \big[kU + (k-1)\big]\eta_\beta \leq D_E\eta_\beta, \tag{38}$$

$$|\widehat{u}_z - u^\star_z| \leq U\eta_\beta. \tag{39}$$

The second inequality uses the preserved first-layer values in (36). The bound in (38) includes all attention leakage to early, semantically invalid checks: their range is bounded even though their interpretation as rule violations need not be valid.

Output error. Equations (30), (38), and (39) imply

$$|\widehat{\xi}_z - \xi^\star_z| \le D_\xi \eta_\beta.$$

Using the global Lipschitz bounds for F and ReLU gives

$$|\widehat{S} - S^\star| \le KBD_\xi \eta_\beta, \qquad |\widehat{C} - C^\star| \le KD_E \eta_\beta.$$

Therefore, for every value token $0 \le v \le B$,

$$|\widehat{\lambda}_v - \lambda^\star_v| \le 2v|\widehat{S} - S^\star| + v^2|\widehat{C} - C^\star| \le M\eta_\beta. \tag{40}$$

This calculation does not require the actual indicator sums or selection gates to be integers.

Completion of Theorem H.1. Choose the common attention scale to satisfy

$$\beta \ge \frac{\log(4NM)}{\Delta_N}. \tag{41}$$

Then $M\eta_\beta \le 1/4$. For the correct value $y = q_{n+1}$, the auxiliary margin is at least one, so

$$\widehat{\lambda}_y - \widehat{\lambda}_v \ge (\lambda^\star_y - \lambda^\star_v) - |\widehat{\lambda}_y - \lambda^\star_y| - |\widehat{\lambda}_v - \lambda^\star_v| \ge 1 - \tfrac{1}{4} - \tfrac{1}{4} = \tfrac{1}{2} \qquad (v \ne y).$$

All constants depend only on p, k, K, N, while the candidate coefficients are stored as fixed weights. Since $\Delta_N > 0$ and $M < \infty$, the scale and all network parameters are finite. The same parameters work simultaneously for every legal prefix of every length $2k \le n \le N$. The residual allocation and FFN counts in Section H.6 prove (24), completing the theorem. ■

H.8 Replacing ReLU with GELU

The activation assumption in Theorem H.1 can be relaxed to GELU without increasing depth, head count, residual width, or FFN hidden width. The modification rescales the FFN parameters and replaces exact intermediate arithmetic with uniformly accurate approximations. The positive logit margin then preserves exact next-token prediction. Throughout this subsection, GELU means the function

$$G(x) = x\Phi(x), \qquad \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-u^2/2}\, du. \tag{42}$$

Lemma H.7 (Uniform approximation by scaled GELU) *For $\gamma > 0$, define $G_\gamma(x) = G(\gamma x)/\gamma$ and $\varepsilon_\gamma = 1/(\gamma\sqrt{2\pi})$. Then*

$$\sup_{x \in \mathbb{R}} |G_\gamma(x) - \mathrm{ReLU}(x)| \le \varepsilon_\gamma. \tag{43}$$

Replacing the parameters of an FFN by

$$(W_1, b_1, W_2, b_2) \longmapsto (\gamma W_1, \gamma b_1, W_2/\gamma, b_2) \tag{44}$$

and using activation G realizes the original FFN with every ReLU replaced by G_γ, with the same matrix dimensions.

Proof. Symmetry of the Gaussian distribution gives

$$|G_\gamma(x) - \mathrm{ReLU}(x)| = |x|\Phi(-\gamma|x|).$$

For $u > 0$, writing $\phi(v) = e^{-v^2/2}/\sqrt{2\pi}$,

$$u\Phi(-u) = \int_u^\infty u\phi(v)\, dv \le \int_u^\infty v\phi(v)\, dv = \phi(u) \le \frac{1}{\sqrt{2\pi}}.$$

Substitute $u = \gamma|x|$; at $x = 0$, the difference is zero. This proves (43). The rescaled FFN is

$$\frac{W_2}{\gamma} G\big(\gamma W_1 h + \gamma b_1\big) + b_2 = W_2 G_\gamma(W_1 h + b_1) + b_2,$$

which proves the implementation claim. ■

Let g_γ and F_γ be the functions in (28) and (29), respectively, with each ReLU replaced by G_γ. The sums of the absolute coefficients of their ReLU terms give the global bounds

$$\begin{aligned} \sup_x |g_\gamma(x) - g(x)| &\le A_g \varepsilon_\gamma, \quad A_g = 4(J+1), \\ \sup_x |F_\gamma(x) - F(x)| &\le A_F \varepsilon_\gamma, \quad A_F = 1 + 2pJ. \end{aligned} \tag{45}$$

Define the finite error constant

$$\Gamma = K\big[B^2 k A_g (2U+3) + 2BA_F + B^2\big], \tag{46}$$

where B, U, J are from (23).

Corollary H.8 (GELU with unchanged architecture sizes) *The conclusion of Theorem H.1, including the resource bounds (24) and margin (25), holds with GELU in place of ReLU. Specifically, use the same construction and rescale both FFNs as in (44), with*

$$\beta \ge \frac{\log(8NM)}{\Delta_N}, \qquad \gamma \ge \frac{8\Gamma}{\sqrt{2\pi}}. \tag{47}$$

All other architectural assumptions in Section H.1 are retained.

Proof. Compare the GELU network to the ReLU network using the same finite attention scale β. Superscripts G and R denote these two networks, respectively. This comparison is separate from the ideal-retrieval comparison in Section H.7.

The first attention layer is identical in both networks, so its retrieved scalar values agree exactly. The first FFN writes only the indicator coordinates. By (45),

$$|e^{\mathrm{G}}_{t,z} - e^{\mathrm{R}}_{t,z}| \le A_g \varepsilon_\gamma \qquad \text{at every position } t \text{ and for every candidate } z. \tag{48}$$

This bound includes early positions whose outputs need not be valid recurrence checks. Both FFNs preserve the shared constant coordinate and all scalar-value coordinates. Since the second-layer queries and keys depend only on that constant, its attention weights are also identical in the two networks. Taking a convex combination cannot increase the maximum coordinate error. Thus the aggregate and candidate-value quantities from (36) satisfy

$$|E^{\mathrm{G}}_z - E^{\mathrm{R}}_z| \le k A_g \varepsilon_\gamma, \qquad u^{\mathrm{G}}_z = u^{\mathrm{R}}_z, \qquad |\xi^{\mathrm{G}}_z - \xi^{\mathrm{R}}_z| \le (U+1) k A_g \varepsilon_\gamma. \tag{49}$$

For the second FFN, compare $F_\gamma(\xi^{\mathrm{G}}_z)$ first to $F(\xi^{\mathrm{G}}_z)$ and then to $F(\xi^{\mathrm{R}}_z)$. Use (45) for the first difference and the global Lipschitz bound $\mathrm{Lip}(F) \le B$ for the second. Applying the analogous argument to $G_\gamma(1 - E^{\mathrm{G}}_z)$ and $\mathrm{ReLU}(1 - E^{\mathrm{R}}_z)$ gives

$$|S^{\mathrm{G}} - S^{\mathrm{R}}| \le K\big[B(U+1)kA_g + A_F\big]\varepsilon_\gamma, \tag{50}$$

$$|C^{\mathrm{G}} - C^{\mathrm{R}}| \le K(kA_g + 1)\varepsilon_\gamma. \tag{51}$$

The linear readout, with $0 \le v \le B$, therefore satisfies

$$|\lambda^{\mathrm{G}}_v - \lambda^{\mathrm{R}}_v| \le 2B|S^{\mathrm{G}} - S^{\mathrm{R}}| + B^2|C^{\mathrm{G}} - C^{\mathrm{R}}| \le \Gamma\varepsilon_\gamma. \tag{52}$$

In particular, we do not assume that the GELU indicators lie in $[0,1]$ or remain exactly binary. Their uniform closeness to the ReLU indicators suffices.

By (40), the ReLU logits differ from the ideal logits by at most $M\eta_\beta$. The scale choices in (47) ensure $M\eta_\beta \le 1/8$ and $\Gamma\varepsilon_\gamma \le 1/8$. Consequently, every GELU logit is within $1/4$ of its ideal value. The ideal margin is at least one, so for $y = q_{n+1}$ and $v \ne y$,

$$\lambda^{\mathrm{G}}_y - \lambda^{\mathrm{G}}_v \ge 1 - 2\big(M\eta_\beta + \Gamma\varepsilon_\gamma\big) \ge \tfrac{1}{2}.$$

The approximation bounds are uniform over real inputs, so one finite choice of β, γ works for every legal prefix with $2k \le n \le N$. Rescaling changes no matrix dimensions or reserved residual coordinates, which proves the same resource bounds. ∎