Sandbox for the Blackbox: How LLMs Learn Structured Data

Bingbin Liu [*]
Harvard University
claraliubb@gmail.com

Ashok Vardhan Makkuva[⋆]
Télécom Paris
ashokevardhan@gmail.com

Jason D. Lee
UC Berkeley
jasondlee88@gmail.com

[Draft; feedback welcome and appreciated!]

Abstract

Large Language Models (LLMs) have been the driving force of the AI revolution, underpinning breakthroughs across a myriad of domains. Yet their success has far outpaced our understanding—leaving them powerful but opaque "black boxes." To bridge this gap, there is a growing scientific interest in demystifying LLMs through *structured sandboxes*—controlled environments such as Markov chains or formal grammars that are simple enough to be mathematically tractable yet powerful to suggest practical interventions. These structured settings reveal what patterns LLMs capture, through which mechanisms, and how LLMs learn them during training. As the field and the literature proliferate, there is a growing need to consolidate these insights into a cohesive understanding. To this end, our article provides a unifying perspective on LLM interpretability through three complementary lenses: representation, optimization, and generalization. By connecting these perspectives across transformers and state-space models, we highlight emerging principles that could guide the design of more transparent, efficient, and reliable language models.

1 Introduction

In recent years, large language models (LLMs) have achieved unprecedented success across various disciplines, including natural language processing [Vaswani et al., 2017], computer vision [Dosovitskiy et al., 2020], and reinforcement learning [Lu et al., 2024]. This success has spurred a flourishing body of research aimed at understanding these models, from theoretical perspectives such as representation [Hahn, 2020, Merrill and Sabharwal, 2022, Liu et al., 2023a] and optimization [Li et al., 2023, Makkuva et al., 2025, 2024], and scientific approaches such as interpretability [Elhage et al., 2021, Olsson et al., 2022, Nanda and Lieberum, 2022]. Such understanding is crucial, given the increasing ubiquity of the models, especially in safety-critical applications [Bai et al., 2022].

A major challenge in understanding language models arises from their complexity and the numerous entangling factors, such as data, architecture details, and optimization algorithms. This motivates the use of *sandboxes*, which are simple abstractions that are designed to capture key aspects of the problem while abstracting away the unnecessary details. The simplicity of sandboxes offers clarity. Predating modern machine learning, simple sandbox models have been adopted in scientific domains for mathematically modeling, such as spin glasses in physics and the Hardy-Weinberg principle in genetics. Specific to complex machine learning systems, the controllable nature of sandboxes make them suitable for diagnoses and stress test purposes, often revealing surprising failure modes and insights. These insights can lead to algorithmic improvements, making sandboxes a cost-effective approach towards principled progress.

This article surveys results following such sandbox approaches. We focus on sandboxes with *structured data*, where we have complete knowledge and control of the data properties. The goal is to provide a unifying perspective on recent advances in the analysis of LLMs, from a representational-cum-learning viewpoint. We

[*]Equal contribution. This writeup is based on the NeurIPS tutorial Sandbox for the Blackbox.

focus on two predominant classes of language: Transformers [Vaswani et al., 2017] and recurrent models such as classic recurrent neural networks [Elman, 1990, Hochreiter and Schmidhuber, 1997] and the more recent state-space models (SSMs) [Gu et al., 2022, Gu and Dao, 2023]. The reminder of this article is organized as follows: Section 3 focuses on representability, Section 4 on optimization landscape and learning dynamics, and Section 5 on generalization properties of LLMs. Section 6 finishes with discussions.

2 Preliminaries

Notation. Scalars are denoted by such italic lower case letters as x, y, Euclidean vectors by bold $\boldsymbol{x}, \boldsymbol{y}$, and matrices by upper case X, Y, etc. $\mathbf{1}$ refers to the all-one vector. For $T \in \mathbb{N}$, $[T] \triangleq \{1, \ldots, T\}$, and for a sequence $(x_t)_{t\geq 1}$, define $x_k^t \triangleq (x_k, \ldots, x_t)$. For $z \in \mathbb{R}$, $\text{sigmoid}(z) \triangleq 1/(1+e^{-z}), \text{ReLU}(z) \triangleq \max(0, z)$, and $\text{softplus}(z) \triangleq \log(1+e^z)$. $\text{Unif}(S)$ denotes the uniform distribution over a set S and $\text{Dir}(\boldsymbol{\beta})$ denotes the Dirichlet distribution with parameter $\boldsymbol{\beta} > 0$. $D_{\text{KL}}(P \| Q)$ denotes the KL divergence between distributions P and Q.

We will discuss two types of language models: Transformers whose computation is parallel across positions, and recurrent neural networks (RNNs) which computes sequentially across positions.

Transformer A Transformer consists of a stack of self-attention layers. For a L-layer Transformer, let X^{l-1} is the output of the $(l-1)_{th}$ layer, then the l_{th} layer for $l \in [L]$ computes a sequence-to-sequence function $f^{(l)} : \mathbb{R}^{d\times T} \to \mathbb{R}^{d\times T} : f^{(l)}(X^{(l-1)}) = f_{\text{mlp}}(f_{\text{attn}}(X^{l-1}))$. f_{attn} denotes the self-attention block $[f_{\text{attn}}(\boldsymbol{x})]_{:,t} = \sum_{i\in[T]} a_{i,t}\boldsymbol{x}_i$, where $a_{i,t} \in [0,1]$ denotes the attention weights with $\sum_{i\in[T]} a_{i,t} = 1, \forall t$. We consider causal attention unless specified otherwise, i.e. $a_{i,t} = 0$ if $i > t$. f_{mlp} denotes the MLP block, typically with ReLU activation and 1 or 2 hidden layers. Residual connections can be added to both f_{attn} and f_{mlp}.

RNN A recurrent neural network $f_{\text{rnn}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ computes a function recursively across time steps: at time t, f_{rnn} updates its hidden state $\boldsymbol{h}_t$ as $\boldsymbol{h}_t = f_{\text{rnn}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$, where $\boldsymbol{x}_t$ is the input at time t. For the purpose of this survey, we treat the implementation of f_{rnn} as a black box.

3 Representability

Understanding the theoretical limits of LLM capabilities helps guide our expectation of their practical performance. These theoretical limits are often referred to as the "representability", "capacity", or "expressivity" of a model. The results aim to provide a mapping between the model sizes and the classes of tasks solvable by the model. Representability results are often stated in two equivalent ways: given a task, what model size is required to ensure the existence a solution to the task; and given a fixed-size model, what (classes of) tasks can be solved. We will use the former, where upper or lower bounds refer to the bounds on the model size. This contrasts to [Strobl et al., 2023], which takes the latter view and instead bounds the class of solvable problems.

While the size of the model is often specified by multiple parameters in practice, we are mainly concerned about the width d and the depth L in theory. The width refers the MLP dimension, as well as attention dimension for Transformers or state dimension for recurrent models. The depth refers to the number of Transformer layers (each consisting of an attention layer and a MLP) or recurrent layers. For attention, the number of heads is simplied as one unless specified otherwise. We will focus on results with a polynomial dependency in both model dimensions d, L and the sequence length T, and finite precisions, which are of better practical relevance than otherwise. Results that are of theoretical significance but are not featured in this article include the universal approximation results in [Yun et al., 2020] have a construction whose depth depends exponentially in d and T, and the Turing completeness result in [Pérez et al., 2021] assumes infinite precision.

In the following, we will discuss common tools used for proving upper (Section 3.1) and lower bounds (Section 3.2), [1] and then discuss applications and implications of these representational bounds (Section 3.3).

[1]We focus on highlighting a few proof techniques that are not limited to Transformers. We refer the readers to [Strobl et al., 2024] for a more

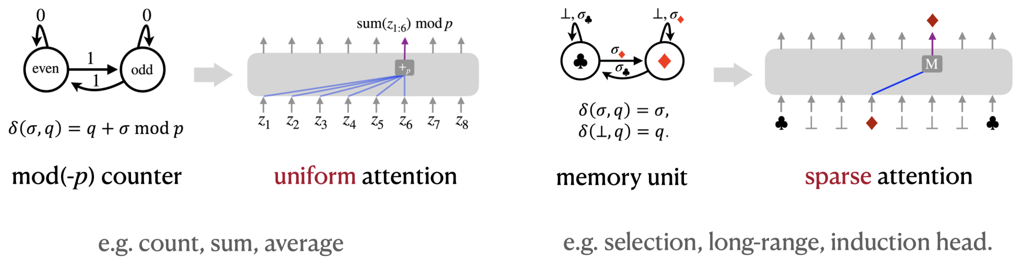

0
0
even
odd
1
1
$\delta(\sigma, q) = q + \sigma \bmod p$
mod(-p) counter
$\text{sum}(z_{1:6}) \bmod p$
z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8
uniform attention
$\perp, \sigma_♣$
$\perp, \sigma_♦$
$\sigma_♦$
$\sigma_♣$
$\delta(\sigma, q) = \sigma$,
$\delta(\perp, q) = q$.
memory unit
M
sparse attention
e.g. count, sum, average
e.g. selection, long-range, induction head.

Figure 1: **Two primitives for Krohn-Rhodes.** Adapted from Liu et al. [2023a].

3.1 Representability upper bounds

Representability upper bounds are typically shown with explicit solution constructions. We will start with case studies on a few widely used sandboxes, then show how we can unify a large group of sandboxes by viewing them as special cases of finite-state automata.

3.1.1 Parity and two modes of attention

Let's warm up with a simple task of *parity*. The input is a binary string $\boldsymbol{x} \in \{0,1\}^T$, and the output y is 0 or 1 depending on whether there are an even or odd number of bit-1 in $\boldsymbol{x}$, i.e. $y = (\sum_{i \in [T]} x_i) \mod 2$. [2] Parity is one of the simplest sequence task and corresponds naturally to the computation in a single Transformer layer: with $\text{poly}(T)$ width and $\log(T)$ precision, the attendion computes the mean (i.e., $f_{\text{attn}}(\boldsymbol{x}) = \frac{1}{T}\sum x_i$), and the MLP computes the mod function (i.e., $f_{\text{mlp}}(z) = (nz) \mod 2$).

What makes Transformer suitable for solving parity is that the attention effectively performs *averaging* and equivalently a scaled *sum*, which is a useful primitive for various tasks such as counting [Bhattamishra et al., 2022, Golovneva et al., 2024], addition [Jelassi et al., 2023, Lee et al., 2023, Shen et al., 2023, McLeish et al., 2024], and capturing the hierarhical structures in languages [Hahn, 2020, Yao et al., 2021, Wen et al., 2023]. Averaging is implemented using **uniform attention**, with attention weights $a_{i,t} = \frac{1}{t}$ for $i \in [t]$. Another attention pattern is **sparse attention**, where $a_{i,t} = 1$ for a single $i \in [t]$ and $a_{i,t} = 0$ otherwise. [3] Sparse attention corresponds to another useful primitive of input-aware *selection* or *copying* [Zhang et al., 2022, Jelassi et al., 2024]. Uniform and sparse attention represent two extreme "modes" of attention serving as useful theoretical abstractions (Figure 1), and attention patterns in practice can be considered as interpolations of the two.

We saw that parity, or more generally counting and selection, are easily implementable with one Transformer layer. In this section, we discuss how to find solutions to a broad class of sequence-to-sequence tasks, which correspond to finite-state automata [Liu et al., 2023a]. Automata model transitions between a set of states given inputs; one can also think of states as nodes in a graph and inputs as specifying which edge to take. Automata capture rich classes of problems associated with sequential reasoning, such regular languages and Markov models[4] which are widely used in reinforcement learning and control.

3.1.2 Automata as sandboxes for sequence-to-sequence tasks

Formally, an automaton $\mathcal{A}$ is defined with $\mathcal{A} := (Q, \Sigma, \delta)$, where Q is the state space, Σ is the input vocabulary, and $\delta : Q \times \Sigma \to Q$ is the transition function. To solve a task formalized by an automaton means to *simulate* the

complete survey connecting Transformers and formal languages.

[2]One can equivalently use $\boldsymbol{x} \in \{\pm 1\}^T$ and $y = \prod_{i \in [T]} x_i \in \{\pm 1\}$. We choose to use 0, 1 since it more naturally maps to attention which computes the weighted sum.

[3]Sparse attention is also known as "hard attention" or "saturated attention", which can be implemented by changing the softmax to the (hard) max function. In case of tie, one can choose to select one or the argmaxes, or put equal attention on the set of argmaxes [Merrill et al., 2022, Hao et al., 2022, Strobl, 2023].

[4]We will discuss more results using Markov models in Section 4.1.

Key properties	Associativity	Solvability (Krohn-Rhodes)	Special cases
Depth	$O(\log T)$	$O_{\|Q\|}(1)$	$O(1)$
Applicable	All $\mathcal{A}$	Sovable $\mathcal{A}$	Commutative $\mathcal{A}$, gridworld

Table 1: **Transformer depth for simulating an automaton** $\mathcal{A}$ for T transitions (Section 3.1.2).

transitions in the automaton, which will be the goal of this section. Note that the definition of an automaton fits naturally to the structure of an RNN, whose hidden states encode automaton states, and the RNN unit is recursively applied in the same way the transition function δ is recursively applied. Such recursive computation means that simulating T transitions corresponds to T sequential steps.

In contrast to RNNs, the computation in Transformers are parallel, making it possible to simulate T transitions in $o(T)$ sequential steps. Moreover, the sequential computation in Transformer happens across layers, hence quantifying the number of sequential steps directly prescribes the Transformer depth. Below we describe three types of such parallel solutions.

$O(\log T)$ **depth by associativity** We first describe a solution that is applicable to simulating transtions of any automaton. The key observation is that simulating transitions effectively computes the composition of sequences of functions, and function composition is associative. This means we can use the good old divide-and-conquer strategy in computer science, and compose T functions in $\log T$ steps. The specific Transformer implemention uses $O(\log T)$ layers, with $O(\log T)$ precision and $O(1)$ width [Liu et al., 2023a]. Beyond simulating automata, this classic idea, known as associative scan [Blelloch, 1990], has also crucially influenced the recent deep learning literature [Merrill and Sabharwal, 2025, Li et al., 2025], including the efficient implementation of state-space models [Gu and Dao, 2023].

$O_{|Q|}(1)$ **depth by Krohn-Rhodes theory** For the subset of automata that are *solvable*, [5] Transformer can implement $O_{|Q|}(1)$-layer solutions, which depends only on the state space size $|Q|$ but not the sequence length T. We can borrow intuition from parity, for which 1 Transformer layer suffices by implementing count-then-mod, regardless of the sequence length. One can generalize the mod-2 for parity to the generic mod-p for any prime p, which is referred to as a "mod counter". Then, similar to how any natural number can be decomposed into a product of primes, any *solvable* automaton can be decomposed (by definition) into a generalized notion of product of mod-p counters.

Such decomposition is formalized by **Krohn-Rhodes theory** [Krohn and Rhodes, 1965], which states that any solvable automaton can be decomposed into two primitives: mod counters, which are implementable by uniform attention, and "memory units", requiring hard selection to simulate, which are implementable by sparse attention. The decomposition is performed on the *transformation semigroup* of an automaton, which is organized in a 2-level hierarchy by the two primitives: the upper level is determined by memory units, which each reduces the state space size by at least 1; the lower leverl is determined by mod counters, which each shrinks the the size of the current transformation (semi)group (upper bounded by $|Q|^{|Q|}$) by at least half. Therefore, the total depth in the hierarchy is upper bounded by $O(|Q|^2 \log |Q|)$. The construction in [Liu et al., 2023a] uses attention width of $2^{\widetilde{O}(|Q|)}$ and MLP width of $|Q|^{\widetilde{O}(|Q|)} + T \cdot 2^{\widetilde{O}(|Q|)}$.

$O(1)$ **depth in special cases** In some special cases, we can further remove the depth dependency on $|Q|$. One example is parity or count-then-mod in general, which is implementable with $O(1)$ layer because the transition function is commutative. For the same reason, any automata with commutative transition functions can be implemented by $O_{|Q|}(1)$ layers. For non-commutative δ, one example is the 1-dimension gridworld [Liu et al., 2023a], which can be considered as addition with over/underflow, a potential scenario when using insufficient precision [Li et al., 2024b]. The idea of the construction is to use $O(1)$ layers to detect positions of over/underflow and $O(1)$ layers to perform normal addition.

[5] Solvability is used in the group theory sense, which for automata refers to the solvability of the associated transformation semigroup. See [Liu et al., 2023a] for details.

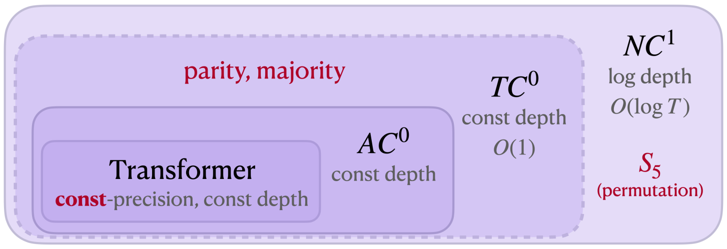


Figure 2: Correspondences of constant-precision constant-depth Transformers and circuit complexity classes, following [Li et al., 2024b]. Solid lines denote strict subset, and dashed lines denote conjectured but unproven strictness.

3.2 Representability lower bounds

Lower bounds are concerned of identifying the minimal size required for any solution be representable. In the following, we will discuss two tools that are typically used for lower bounding the depth and width of a network.

Remark: A lower bound can be conditional or unconditional. A conditional lower bound relies on statements that are unproven but generally hypothesized to be true or unconditional, whereas an unconditional lower bound uses only proven results. We will distinguish between assumptions and proven statements when discussing the results.

3.2.1 Depth lower bound via circuit complexity

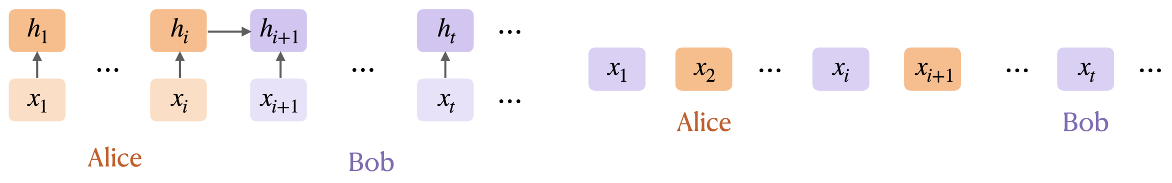
The general recipe of using circuit complexity for depth lower bound consists of the following steps. First, identify two circuit classes $\mathcal{C}_1$, $\mathcal{C}_2$ separated by depth, such that $\mathcal{C}_2 \subset \mathcal{C}_1$ and $\mathcal{C}_2 \neq \mathcal{C}_1$; the strictness of such containment is typically where unproven conjectures are introduced. Then, show that the task of interest $\mathcal{A}$ is in $\mathcal{C}_1$ but not in $\mathcal{C}_2$; one choice is to make $\mathcal{A}$ the hardest task in $\mathcal{C}_1$, which, by the assumption that $\mathcal{C}_2 \neq \mathcal{C}_1$, would imply that $\mathcal{A} \notin \mathcal{C}_2$. Finally, show that a Transformer of a certain depth is contained in $\mathcal{C}_2$, hence cannot represent the task of interest; this can be established by "compiling" a Transformer into a circuit [Merrill and Sabharwal, 2022].

To give a concrete example, let's revisit the automata discussion in Section 3.1.2, where we provided $O_{|Q|}(1)$-depth construction only for solvable automata, but not for any automaton. The reason is that $O_{|Q|}(1)$ depth cannot represent certain automata according to a widely believed conjecture. Specifically, the two classes seperated by depth are $\mathcal{C}_1 = \mathrm{NC}^1$ (i.e. Nick's class (NC) of log depth), [6] and $\mathcal{C}_2 = \mathrm{TC}^0$ (i.e. threshold circuits (TC) of constant depth). The automaton $\mathcal{A}$ is chosen to be the symmetric group of order 5, denoted as S_5. S_5 is known to be NC^1-complete and conjectured to be outside of TC^0. Now it remains to show that constant-depth Transformers are contained in TC^0, which has been established by [Merrill and Sabharwal, 2022].

Results in [Merrill and Sabharwal, 2022] are proven for $\log T$ precision, which is a common setup since at least $\log T$ precision is required for representing positions. However, in practice we often operate with constant precisions, which further limits the representational power of Transformers. In particular, [Li et al., 2024b] shows that with constant precision, constant-depth Transformer lies within AC^0, which is known to be a proper subset of TC^0 that cannot even compute simple functions such as parity and majority (Figure 2).

Remark: While depth lower bounds are of great theoretical value, it would be too persimistic to treat stataements such as "practical Transformers are in AC^0" as a practical takeaway. It is worth noting that these circuit lower bounds are *asymptotic*, where the counterexamples are typically established with sufficiently large input sizes. Problem instances of practice interest may be well within the theoretical limit. Nevertheless, understanding the effect of depth can offer insights on practical concerns such as generalization, which we will discuss briefly in Section 5.

[6] Here NC denotes the type of gates in the circuits, and the superscript (e.g. 1 or 0) denotes the depth in terms of the power of log (of the number of circuit inputs).



(a) RNN (b) Transformer

Figure 3: Illustrations of the communication protocols in RNN and Transformers. The communicating parties are *(Top)* two segments of the sequence for an RNN, and *(Bottom)* two subsets of the sequence positions for a Transformer.

3.2.2 Width lower bound via communication complexity

For width lower bounds, a common strategy is to argue from an information perspective by leveraging communication complexity [Sanford et al., 2023, 2024b, Peng et al., 2024, Chen et al., 2024a, Jelassi et al., 2024, Arora et al., 2024]. Communication lower bounds are on the number of bits that two or more parties need to communicate in order to jointly solve a task. To lower bound a width of interest, we form the task as a communication problem by making the width the communication bottleneck, and different parts of the network as connunicating parties. In the following, we discuss this how this can be done for RNNs and Transformers respsectively (Figure 3).

RNN hidden size An RNN processes information sequentially and recursively. Unrolling its computation graph into a line graph (Figure 3a), we can treat one section of the line as one party, which we refer to as Alice, and the other section as another party, which we refer to as Bob. That is, Alice holds positions 1 through t for some t, and Bob holds positions $t+1$ onwards. The hidden state naturally becomes the bottleneck: recall that an RNN computes $h_{t+1} = f_{\text{rnn}}(h_t, x_{t+1})$. In order for the two parties to perform the full computation of f_{rnn}, Alice needs to send h_t to Bob, hence the amount of information communicated is the size of h_t, i.e. its dimension times its precision.

Transformer hidden size For Transformers, the communications happen in a pairwise fashion between the positions. The two communication parties can be defined to be one subset of the positions and its complement. Suppose Alice holds a set of positions S and Bob holds the other positions $\bar{S}$, as illustrated in Figure 3b. Then, in order for Bob to compute the output of a position t, he sends the query vector to Alice, who will compute and send back the attention-score-aggregated value vector; that is,

$$\frac{\sum_{i\in\bar{S}} \exp(\langle \boldsymbol{q}_t, \boldsymbol{k}_i\rangle)\boldsymbol{v}_i + \underbrace{\sum_{j\in S} \exp(\langle \overbrace{\boldsymbol{q}_t}^{\text{Bob sends to Alice}}, \boldsymbol{k}_j\rangle)\cdot \boldsymbol{v}_j}_{\text{Alice sends to Bob}}}{\sum_{i\in\bar{S}} \exp(\langle \boldsymbol{q}_t, \boldsymbol{k}_i\rangle) + \sum_{j\in S} \exp(\langle \boldsymbol{q}_t, \boldsymbol{k}_j\rangle)}. \tag{1}$$

Hence the amount of information communicated is the precision times the dimension of $\boldsymbol{q}$ and $\boldsymbol{v}$.

One notable example of this communication perspective is the relationship between the *Massively Parallel Computation* (MPC) model [Karloff et al., 2010] and Transformers [Sanford et al., 2024b]. The MPC framework studies communication across parallel machines, typically assuming unbounded local computation but bounded local memory per machine, with communication occurring in synchronized rounds. For Transformers, each sequence position can be viewed as a local machine: the MLP provides local computation, and attention implements a communication step as described earlier. In particular, Sanford et al. [2024b] show the bi-directional simulation between Transformer and MPC, where the number of MPC rounds and Transformer layers are related linearly up to model-dependent overheads. Given the well-studied connection between MPC and graph algorithms, this correspondence helps characterize Transformers' ability to solve a range of graph tasks, and it enables fine-grained analysis of the role of width [Sanford et al., 2024b, Yehudai et al., 2025].

To summarize, the depth lower bounds, established using tools from circuit complexity, apply to Transformers, whereas the width lower bounds derived from the communication lens apply to both Transformers and RNNs, with an additional connection between Transformer and MPC. We will see applications of these lower bounds in Section 3.3.

3.3 Applications of representability results

While representability results are theoretical, they provide conceptual understanding and insights for architectural improvements. We discuss some examples below.

3.3.1 Benefit of depth

Our first example is to understand the role of depth. While a smaller depth is often preferrable in practice for efficiency considerations, depths are important for representability in two senses: there are scenarios where depth is required for *any* solution to exist, as we saw for S_5 (Section 3.2.1), and where depth is required for an *efficient* solution to exist, which we will discuss next. Consider the task of *induction head* [Elhage et al., 2021, Olsson et al., 2022], which are useful for conditional copying a simple sandbox that is widely useful. Several work have shown that induction heads can be implemented with 2 Transformer layers of $d = O(1)$ width, $H = O(1)$ heads, and $p = O(\log T)$ precision [Zhang et al., 2022, Bietti et al., 2023, Liu et al., 2023b]. [7] While a single Transformer layer also suffices, [Sanford et al., 2024a] shows that there will be a width lower bound of $d \cdot H \cdot p = \Omega(T)$. The proof uses the communication complexity lower bounds discussed in Section 3.2.2, where the induction head task is reduced to the classic INDEX problem [Kushilevitz and Nisan, 1997].

Remark: So far we have connected depth to circuit complexity, though there are other complexity notion. A closely related alternative is formal logic [Chiang et al., 2023, Barceló et al., 2023, Yang and Chiang, 2024, Yang et al., 2025], which can provide a more fine-grained hierarchy than circuit complexity. In particular, [Yang et al., 2025] shows Transformers with fixed-precision attention are equivalent to a programming language known as C-RASP [Yang and Chiang, 2024], where more depth provides more expressive power. We refer to the survey [Strobl et al., 2024] for more discussions on formal logic. Another complexity measure is the separation rank of the tensor formed by the model outputs, which has been used in [Levine et al., 2020] to study depth-width tradeoff. They consider a simplified Transformer whose model outputs are polynomials of the inputs, and show that the model width and depth impact the separation rank of the output tensor differently; in particular, depth can more efficiently improve the separation rank provided that the width is sufficient.

3.3.2 Architectural comparison

Representation power is one aspect of the architectural comparison between Transformers and recurrent networks. There are tasks that are more efficiently representable with one than the other. For tasks that are more efficiently representable by Transformers, one common idea is to make the task memory-heavy, so that RNN's hidden states will be a memory bottleneck unless using a large size as we discussed regarding width lower bounds (Section 3.2.2). Examples include retrieval, copying, associtiative recall, or (k-hop) induction heads [Arora et al., 2024, Bhattamishra et al., 2024, Jelassi et al., 2024, Sanford et al., 2024a, Wen et al., 2025]. For the reverse direction, i.e. when showing RNN is better than Transformer, we will consider a single forward pass of the Transformer as otherwise we can use Transformer in an auto-regressive fashion to simulate an RNN. In this case, Transformer is often limited in representation power due to limited depth such as the case for S_5 and bounded Dyck language [Merrill and Sabharwal, 2022, Liu et al., 2023a, Bhattamishra et al., 2024] (Section 3.2.1).

3.3.3 Architectural improvements

Given the bottlenecks of memory for RNNs and recurrent depth for Transformers, there are natural interventions one can take to overcome these limitations. Below we discuss two examples.

[7] The idea is that the first layer creates pairs of $(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ for each position t, and the second layer retrieves the correct pair by matching the first element of each pair.

Improving effective depth for Transformers For Transformers, we can increase its effective depth by using it generatively, such as in a chain-of-thought (CoT) [Wei et al., 2022] fashion which several works have shown to be effective [Feng et al., 2023, Malach, 2023, Merrill and Sabharwal, 2023, Li et al., 2024b]. As an intuition, a circuit of M gates can be simulated by $O(M)$ steps of CoT: at each step, the attention collects inputs to the current gate, and the MLP computes the gate. Formally, [Merrill and Sabharwal, 2023] shows that a log number of CoT steps enlarges the representable complexity class from within TC^0 to within log-space, and a linear number of CoT steps suffices to represent all regular languages. [Li et al., 2024b] draws a more careful distinction by additionally considering the effect of precision and embedding dimension. Notably, using a constant precision places Transformer under AC^0 as mentioned in Section 3.2.1, and using $\log T$ CoT steps improves over no CoT only when the embedding dimension is also $\log T$, and has no effect when the dimension is $\text{poly}(T)$.

One may wonder how CoT compares to directly increasing the number of layers. From the representability perspective, we can compare two aspects. The first aspect is the number of sequential steps required to achieve a desired capacity, which impacts the sequential runtime. Increasing depth is preferrable in this case: [Merrill and Sabharwal, 2025] shows that there exists tasks (in particular, graph connectivity) that can be solved with $O(\log T)$ depth, but not with $O(\log T)$ CoT steps. The second aspect is the notion of *uniformity*, that is, whether the same solution is applicable across varying T. This is of practical relevance, for example, when considering length generalization [Anil et al., 2022, Jelassi et al., 2023, Zhou et al., 2024a,c, Huang et al., 2024, Golowich et al., 2025, Chen et al., 2025]. CoT and increasing depth can both achieve uniformity; in particular, the constructions in [Merrill and Sabharwal, 2025] are uniform. [8]

Improving memory for RNNs As discussed in Section 3.2.2, recurrent networks can suffer from limited memory due to an insufficient hidden state size, which can not be addressed by CoT. Instead, the memory can be increased by integrating with external retrieval mechanism or attention layers. [Wen et al., 2025] shows that as little as 1 attention layer suffices to enable RNNs to simulate polynomial-time Turing machines. This is consistent with the empirical success of hybrid models, where attention layers are inserted sparsely inbetween recurrent layers; for instance, Jamba [Team et al., 2024] uses a recurrent-to-attention ratio of 7-to-1. Since attention layers have a heavier (GPU) memory footprint than RNNs, such sparse use of attention layers is favorable and a promising approach for balancing (GPU) memory usage and model performance [Team et al., 2024, Glorioso et al., 2024, Ren et al., 2024, Wang et al., 2024b, Waleffe et al., 2024, Blakeman et al., 2025].

4 Optimization & Learning Dynamics

As Section 3 highlights, while representational results characterize what solutions a model *can* express, they do not explain *whether* or *how* such solutions are discovered during training. Given that neural network optimization is highly non-convex, understanding the optimization landscape and learning dynamics of language models is therefore essential to bridge the gap between expressivity and practical learnability. To this end, we examine the learning dynamics of Transformers trained on structured data, that replicate many intriguing behaviors observed in practice.

Specifically, we consider a variety of structured inputs, such as Markov chains/n-grams and their anciliary processes, which received attention the most in the literature, followed by topic models and compositional inputs. For the exposition, we focus primarily on the interesting phenomena captured by these various setups and emanating insights, providing requisite mathematical preliminaries along the way but we avoid stating results that are too technical in nature.

4.1 Sequential structure: Markov chains and n-grams

We start with Markov processes. Markov processes, or equivalently n-grams, are arguably the simplest yet fundamental models for natural language [Shannon, 1948, Merialdo, 1994], and have a long and well-established history. Formally, a sequence $(x_n)_{n \geq 1}$ in a discrete vocabulary $\mathcal{X}$ is called a k^{th}-order Markov chain (or equivalently a $(k+1)$-gram) if the conditional distribution of the next symbol x_{n+1} depends solely on the previous

[8]In contrast, the construtions in [Liu et al., 2023a, Li et al., 2024b] are non-uniform.

k symbols, i.e. $(x_n, \ldots, x_{n-k+1})$, for all $n \geq 1$. For example, when $\mathcal{X} = \{0,1\}$ and $k = 1$, we have a first-order Markov chain with binary states where x_{n+1} only depends on x_n. In this case, the dynamics of the process is fully captured by the transition probabilities between the states 0 and 1, as illustrated in Fig. 4 with transition probabilities p and q in $[0,1]$. Similarly, we can define a state transition matrix for a k^{th}-order Markov process, based on 2^k possible states the past k symbols can take. If all the sequences are sampled according to the same transition kernel, they are referred to as *global Markov chains*.

$$\begin{tikzcd} 0 \arrow[loop left, "1-p"] \arrow[rr, bend left, "p"] & & 1 \arrow[loop right, "1-q"] \arrow[ll, bend left, "q"] \end{tikzcd}$$

$$\Updownarrow$$

$$(x_{n+1} \mid x_n)_{n \geq 1} \sim \boldsymbol{P}(p,q) \triangleq \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix},$$
$$\boldsymbol{P}_{ij} = \mathbb{P}\big(x_{n+1} = j \mid x_n = i\big), \;\; i,j \in \{0,1\}.$$

Figure 4: State transition diagram and Markov kernel for a first-order Markov chain $\boldsymbol{P}(p,q)$ with flipping probabilities $\boldsymbol{P}_{01} = p$ and $\boldsymbol{P}_{10} = q$.

On the other hand, *in-context Markov chain* [9] refers to a sequence sampled from a Markov kernel $\boldsymbol{P}$, which itself is drawn from a prior distribution, typically a Dirichlet with parameter $\mathbf{1}$ [Norris, 1997]. Namely, the transition $\boldsymbol{P}_{i_1^k,:} \triangleq P(\cdot|x_1 = i_1, \cdots, x_k = i_k)$ is sampled independently and uniformly on the $|\mathcal{X}|$-dimensional simplex $\Delta_1^{|\mathcal{X}|}$, for each tuple $(i_1, \cdots, i_k) \in \mathcal{X}^k$. Thus no two sequences (almost surely) share the same transition matrix $\boldsymbol{P}$. In this setup, successful prediction requires the model to infer the sequence-specific transition probabilities directly from the observed context, enabling in-context estimation.

Capitalizing on the Markovian structure, we now consider the intricate interplay between the order of the process and model depth to reveal many interesting properties exhibited by Transformers, starting with single-layer models.

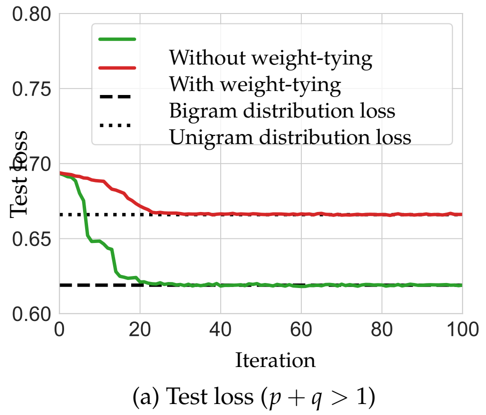
4.1.1 Single-layer Transformers on first-order Markov chains: A suprising failure mode

Modeling the input as a first-order (global) Markov chain as in Fig. 4, Makkuva et al. [2025] analyze the loss landscape of a single-layer Transformer and establish a surpising result that even in this simplest setting, Transformers could get stuck at local minima depending on their weight-tying and Markov state-switching probabilities, $p + q$, also known as the *switching factor*. Fig. 5 illustrates this phenomenon.

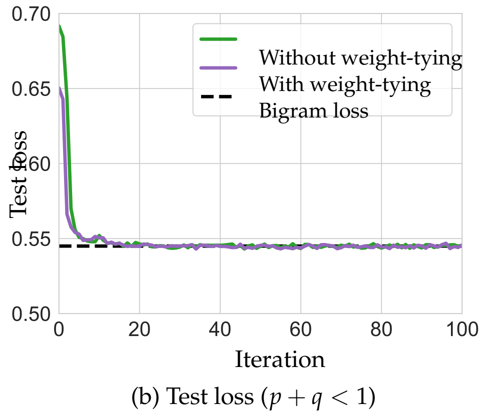
More precisely, they theoretically demonstrate with explicit construction that when Markovian switching $p + q$ is greater than 1, weight tying can induce bad local minima on the loss surface, whereas removing the tying, the same minima become saddle points, potentially allowing the model to escape to global minima. Fig. 5a highlights this for $p = 0.5$ and $q = 0.8$. Here the local minima correspond to the model capturing only the unigram/marginal distribution, $\mathbb{P}(x_{n+1} = 1)$, as opposed to the ground-truth Markov kernel $\mathbb{P}(x_{n+1} = 1 \mid x_n)$, which corresponds to the global minimum.

While these optimization results provide insights into the static loss landscape picture, they do not characterize how Transformers trained via gradient-descent and its variants converge to these critical points. Addressing this, for the same setting as above, a follow-up work [Makkuva et al., 2024] studies the gradient-flow dynamics for weight-tied Transformers and unravels an intricate interplay between parameter initialization and the Markovian

[9]We will discuss in-context learning in detail in Section 4.1.2.

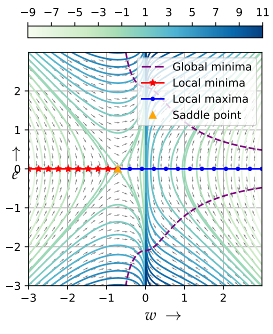


(a) Test loss ($p + q > 1$)

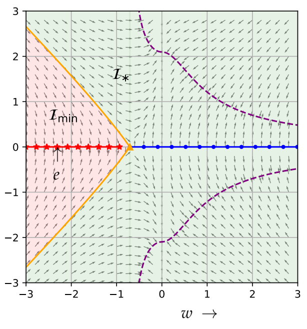


(b) Test loss ($p + q < 1$)

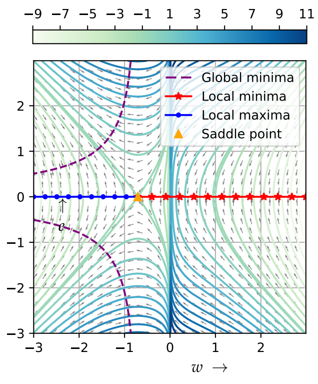
Figure 5: **Effect of weight tying on test loss for singe-layer Transformers trained on first-order Markov chains.** For (a): $p = 0.5, q = 0.8$. With weight tying, the loss converges to a local minimum, whereas without it goes to a global minimum. For (b): $p = 0.2, q = 0.3$. The test loss always converges to a global minimum.



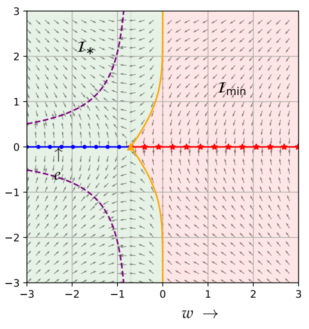
(a) Gradient flow ($p + q < 1$)



(b) Convergence basin ($p+q<1$)



(c) Gradient flow ($p + q > 1$)



(d) Convergence basin ($p+q>1$)

Figure 6: **Gradient flow dynamics and initialization effect for single-layer Transformers.** (p, q) are Markov switching probabilities, and (e, w) are the embedding and weight parameters. (a), (c): The flow is aligned along energy contour lines, converging to local or global optima. (b), (d): $I_\star$ is the basin of convergence for global minima, $I_{\min}$ for the local minima, and yellow asymptotes for the saddle point. Notice the contrasting behavior for Gaussian initialization around origin for $p + q \lesseqgtr 1$.

switching factor $(p + q)$ in determining the overall learning dynamics and convergence to local or global minima, illustrated in Fig. 6.

In particular, we observe that when $p + q < 1$, standard Gaussian initializations around the origin, as is typically done in practice, converge to a global minimum, theoretically corroborating the empirical evidence in Fig. 5b. On the other hand, when $p + q > 1$, we notice that the origin instead falls in the basin of attraction for the local minima, enabling the model to get stuck there, in sync with Fig. 5a.

In summary, these results highlight the complex interplay between the input Markovian properties, model initialization, and its architecture even for the toy setting of single-layer Transformers on first-order Markov chains.

4.1.2 Two-layer Transformers on bigrams: emergence of in-context learning and induction heads.

While single-layer Transformers illuminate the challenges of optimization even in simple Markovian environments, increasing model depth unlocks qualitatively richer behaviors—most notably *in-context learning* (ICL)

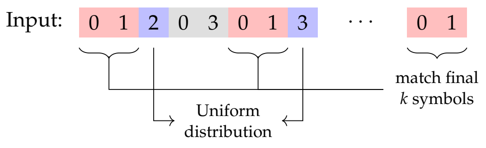


Figure 7: **Conditional k-gram model/k^{th}-order induction head for $k = 2$.** The conditional k-gram is the in-context estimate of the Markov process and is realized in two steps. The first step is to find the locations in the sequence (marked red) which match the final k symbols. The conditional k-gram model returns the uniform distribution over the next symbol at these locations (marked blue).

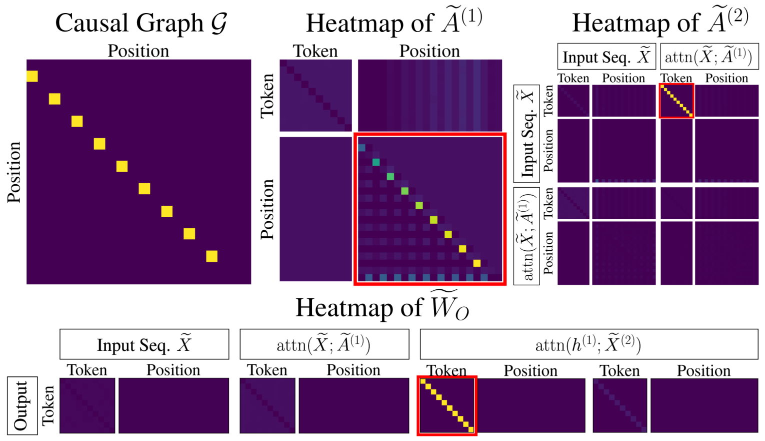


Figure 8: **The weights of a trained disentangled Transformer:** The causal graph $\mathcal{G}$ encodes the Markovian structure. $\widetilde{A}^{(1)}$ and $\widetilde{A}^{(2)}$ denote the attention matrices of first and second layers respectively, whereas $\widetilde{W}^{(O)}$ is the output weight matrix. The highlighted block in $\widetilde{A}^{(1)}$ converges to the adjacency matrix of the causal graph $\mathcal{G}$, and the highlighted blocks in $\widetilde{A}^{(2)}, \widetilde{W}^{(O)}$ converge to the identity matrix. Adapted from Fig. 2 in [Nichani et al., 2024].

[Olsson et al., 2022]. ICL is a hallmark feature of Transformers, referring to the ability to infer new patterns or rules directly from the input context, without any parameter updates. For instance, given a sequence of input–output pairs $[x_1, f(x_1)], [x_2, f(x_2)], \ldots, [x_n, f(x_n)]$, the model must learnt to predict $f(x_{\text{test}})$ purely from the observed pairs, effectively learning "within" the prompt .

A key circuit enabling this behavior in Transformers is the *induction head*—a pair of attention heads that together implement a copy-and-compare operation. More formally, given an input sequence $(x_1, \ldots, x_T) \in [\mathcal{X}]^T$ of length T, the induction head implements a conditional 1-gram model, defined as $\widehat{\Pr}_1(x \mid x_1, \cdots, x_T) \triangleq \left(\sum_{n=2}^{T} \mathbb{I}(x_n = x, x_{n-1} = x_T)\right) / \left(\sum_{n=2}^{T} \mathbb{I}(x_{n-1} = x_T)\right)$, which is defined only so long as the denominator is non-zero. A conditional k-gram is defined likewise, with its counterpart termed a k^{th}-order induction head. Figure 7 illustrates this.

A Transformer with two attention layers can implement this: the first layer learns the attention pattern $\text{att}^{(1)}_{n,i} = \mathbb{I}(i = n - 1)$, i.e. "copy" operation, thereby allowing the model to capture information about the symbol at position $n - 1$ in the embedding vector at time n, implicitly encoding the causal structure of a bigram. In the second layer, the attention layer picks out those indices n where $x_{n-1} = x_T$, the final symbol in the sequence, i.e. "compare" operation. At these positions, since $x_{n-1} = x_T$, one would expect that the next symbol x_n is a good predictor of x_{T+1}, and the model uses this information to predict the next symbol x_{T+1} according to its conditional empirical estimate, $\widehat{\Pr}_1(x_{T+1}|x_1, \cdots, x_T)$, i.e. the conditional 1-gram model. It is well known that the conditional k-gram with Laplace smoothing corresponds to the Bayes optimal estimate of the next symbol probability, when the data is drawn from an in-context Markov chain.

Shedding light on ICL and induction heads vis-a-vis Transformers, [Bietti et al., 2023] is one of the earliest works providing empirical and theoretical insights. Using a synthetic bigram dataset with both global and in-context Markov chains, they showed that while one-layer Transformers fail to predict in-context bigrams, adding a

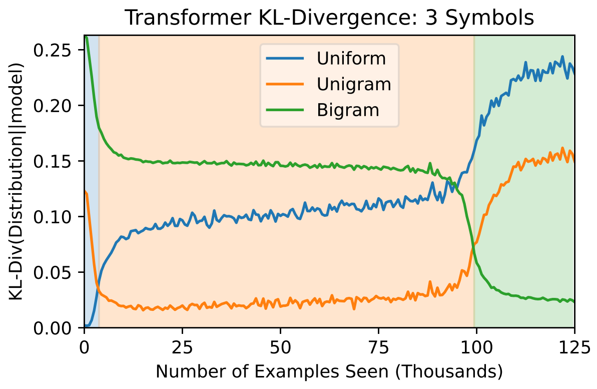


Figure 9: **Phase transitions of Transformers while learning in-context Markov chains:** KL divergence of a Transformer's output distribution to several well-defined strategies over the course of training. The model passes through three stages: (1) predicting a uniform distribution, (2) predicting based on in-context unigram statistics, (3) predicting based on in-context bigram statistics. Adapted from Fig. 1 in Edelman et al. [2024].

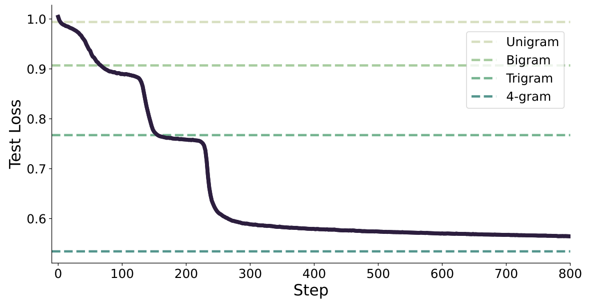


Figure 10: **The stage-wise behavior of the test loss during training for the 4-gram language model:** The dashed lines represent the cross entropy loss of the k-gram estimators for $k = \{1, 2, 3, 4\}$. The plateaus of the test loss overlap with the losses of k-gram estimators. Adapted from Fig. 1 in Varre et al. [2025].

second layer enables the formation of induction heads.

Building on this foundation, Nichani et al. [2024] gives a more rigorous account of how two-layer Transformers acquire in-context learning capabilities from scratch when trained via gradient descent, highlighted in Fig. 8. Specifically, they consider a family of in-context Markov chains with the prior satisfying some standard regularity conditions and demonstrate that a variant of two-layer attention-only Transformer, the *disentangled Transformer*, trained on such data learns to recover these local Markovian dependencies through the emergence of induction heads. Specifically, the first attention layer learns to capture pairwise dependencies between consecutive tokens—capturing the underlying Markovian structure—while the second layer uses this information to perform in-context prediction of future tokens, showing that the induction head arises naturally from the optimization process.

While these results focus on first-order Markov sources with a single attention head per layer, Chen et al. [2024b] extend this line of work to n-gram Markov chains using two-layer Transformers with multi-headed attention, initialized according to a symmetry breaking configuration between attention heads and Markov parents. They show that such models develop a *generalized induction head* capable of capturing higher-order dependencies, further illustrating how structured learning dynamics emerge naturally through gradient-based optimization.

4.1.3 n-grams: Stage wise learning dynamics of Transformers.

In the earlier sections, we saw how gradient-based training enables Transformers to learn induction heads at convergence. Moving beyond the end behavior, Edelman et al. [2024] study how this process unfolds over the course of training. Focusing on bigrams, they empirically reveal that Transformers undergo distinct *phase transitions* when learning in-context Markov chains—marked by abrupt drops in loss between phases. Each phase corresponds to learning models of increased complexity—first unigrams, then bigrams. Fig. 9 illustrates this.

Consolidating these empirical findings, Varre et al. [2025] theoretically characterize stagewise learning in (disentangled) Transformers trained on in-context n-grams (Fig. 10). In particular, they show that the cross-entropy loss landscape contains multiple plateaus corresponding to *sub-n-gram* models, each representing near-stationary points of the population loss. As training progresses, model's learning trajectory naturally pauses at these plateaus of vanishingly small gradient, which represent mastery of lower-order dependencies, before eventually escaping to learn more complex ones, triggering discrete *phase transitions* [Varre et al., 2025].

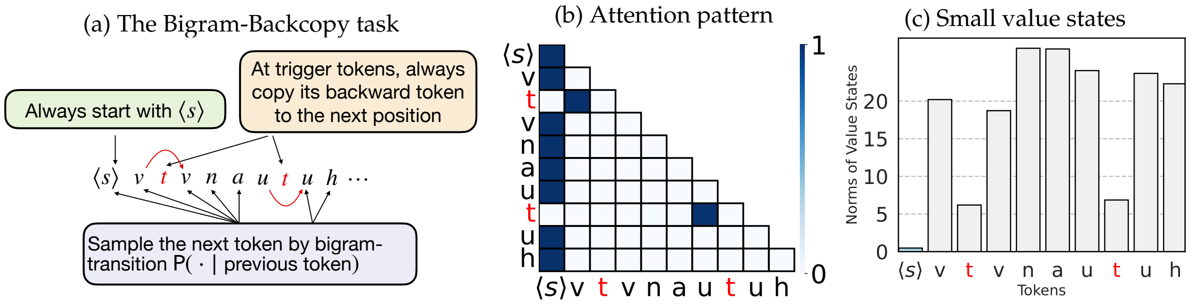


Figure 11: **Bigram-Backcopy task and extreme token phenomena.** *Left (a)*: The data generation procedure for the Bigram-Backcopy task. Here 't', 'e', and the space character (' ') are trigger tokens. The BB task samples bigram transitions for non-trigger tokens and backcopies for trigger tokens. *Middle (b)*: The attention map of a given prompt. Trigger tokens are marked in red. The attention head at non-trigger tokens is dormant and displays attention sinks. *Right (c)*: The value state norms for the prompt. The ⟨s⟩ token has the smallest norm. Adapted from Fig. 2 in Guo et al. [2024].

4.2 Beyond Markov

While Markovian settings provide a principled foundation for understanding the emergence of in-context mechanisms, real-world language often exhibits richer dependencies and mixed causal structures. A series of recent works extend this line of inquiry beyond simple Markov chains and n-grams, revealing new algorithmic behaviors and intriguing phenomena in Transformers.

Along this vein, Wang et al. [2024c] build upon Bietti et al. [2023] to consider a mixed-target, composed of a global 4-gram and an in-context 2-gram component. Leveraging this controlled setting for a precise analysis of the training dynamics, they reveal an abrupt transition from a *lazy regime*, capturing 4-gram models, to a *rich regime*, where induction heads emerge to capture in-context dependencies. Specifically, they show that learning progresses through four phases: partial learning of the 4-gram, induction-head plateau, induction-head formation, and final convergence, showcasing a sharp transition from 4-gram to induction head.

Further offering a novel perspective on ICL, Park et al. [2024] argue that ICL is not a single, monolithic capability but an emergent outcome of multiple competing mechanisms. To support this view, they introduce a unified framework based on a finite mixture of Markov chains, which reproduces most known ICL phenomena, including its transient nature and sensitivity to data diversity and context length. Within this controlled setup, they show that a model's behavior can be decomposed into a small set of algorithmic modes combining *fuzzy retrieval* versus *inference* strategies with either unigram or bigram context statistics. These modes engage in *competition dynamics*, where the dominant algorithm depends on factors such as training duration, model scale, and context size. As these factors vary, models transition—often sharply—between distinct algorithmic regimes, making ICL's behavior inherently phase-like and setting limits on the generality of conclusions drawn from any single configuration.

Outside the realm of ICL dynamics, Guo et al. [2024] investigate the emergence of *extreme-token phenomena* in Transformers—systematic behaviors where certain "sink" tokens attract disproportionately large attention weights (attention sinks), exhibit vanishing value activations (value-state drains), and display high residual norms (residual-state peaks). To elucidate the mechanisms behind these phenomena, they introduce the Bigram-Backcopy (BB) task, a synthetic data-generating process that combines bigram transitions with occasional backcopy, when specific trigger tokens are encountered, enabling the model to alternate between Markovian prediction and direct token copying. Fig. 11 illustrates this. Training small Transformers on this setup, they uncover an *active–dormant mechanism* in which attention heads become sinks for specific input domains while remaining inactive elsewhere. Their theoretical analysis reveals that a *mutual reinforcement mechanism* between attention and value pathways drives the formation and stabilization of these sinks, giving rise to a self-sustaining "extreme-token phase." Notably, similar active–dormant patterns and reinforcement dynamics are observed in pretrained LLMs such as Llama and OLMo, suggesting that the BB framework captures core aspects of how such pathological behaviors arise during large-scale pretraining. Building on these insights, they propose strategies to mitigate them, including replacing softmax with ReLU and Adam with SGD.

Finally, while not focused on learning dynamics, D'Angelo et al. [2025] and Zhou et al. [2024b] examine Transformer representational capacity in extended Markovian settings—the former through interleaved Markov chains, leading to the emergence of *selective induction heads* that infer which chain to attend to, and the latter through *variable-order Markov chains (VOMC)*, showing that Transformers can in-context learn to compress VOMC, even matching or surpassing Bayesian-optimal algorithms like Context Tree Weighting (CTW) through hybrid attention–counting mechanisms.

4.3 Semantic structure: topic models

Moving beyond Markovianity, we next turn to the semantic structure that characterizes natural language, where sequences are shaped by latent topics and conceptual groupings rather than purely local statistical dependencies. Topic-structured sandboxes provide an insightful lens into understanding how Transformers internalize and exploit semantic regularities, shedding light on their ability to learn such structures purely from training.

1) How Transformers learn topic structure. Li et al. [2023] consider a Latent Dirichlet Allocation (LDA)-generated input data as a sandbox [Blei et al., 2003], inspired by the observations that a pretrained BERT model exhibit topical structure in their token embeddings, where the embeddings belonging to the same topic exhibit more cosine similarity than those that of different topics (Li et al. [2023], Figure 1-3). Using this topic model distribution to study the learning dynamics of a single layer Transformer, they show that topic structure can be encoded both in the embedding layer and in the attention mechanism of the network. Interestingly, even if one of these components is frozen, they discover that the other can compensate for it.

More formally, they consider the following topic model, which is a special of the classical LDA: let there be T topics and Tv words, with each topic owning a disjoint vocabulary of size v. A document $\boldsymbol{w} = (w_1, \ldots, w_N)$ is generated by first selecting τ distinct topics $t_1, \ldots, t_\tau \subset [T]$, then for each position $n \in [N]$, sampling a topic $t \in \{t_1, \ldots, t_\tau\}$ and drawing w_n uniformly from its vocabulary $\{(t-1)v+1, \ldots, tv\}$. Thus each word maps to exactly one topic and topics have non-overlapping vocabularies, yielding a clean ground-truth semantic structure for analysis. For the Transformer architecture, they consider a single self-attention layer without the MLP, residual connection, and normalization layers. For training, they consider a masked language modeling objective where the Transformer is fed as input the masked document $\widetilde{\boldsymbol{w}}$, obtained by randomly masking words in $\boldsymbol{w}$ according to some predefined probabilities, and is tasked with predicting the original words at these masked positions. When trained on this objective, they first show that even if the self-attention is frozen to uniform weights, the embedding layer can encode the topic structrue, i.e. embeddings corresponding to same-topic tokens are more similar than those of different topics.

On the other hand, when the encoding layer is frozen to be one-hot embeddings, they demonstrate that self-attention can also perform the heavylifting in learning the topic-modeling distribution, by encoding the topic structures. To this end, they analyze a two-stage optimization process of self-attention, where in Stage 1 the value matrix is trained to optimum, and in Stage 2 the (key, query) matrices are trained freezing the value matrix at this optimal value (Li et al. [2023], Figure 4). Whereas in Stage 1, they theoretically demonstrate that the model predicts masked words by using only unmasked words from the same topic, effectively aggregating topic information to infer the missing tokens, in Stage 2, they show that attention weights corresponding to same-topic tokens are higher on average than those of different topics, again reflecting the topic structure.

In summary, these results highlight how Transformers learn topic structure in the LDA setting, and building upon them, Li et al. [2023] further demonstrate that the same patterns—encoding of topic information in both embeddings and attention—also emerge on real Wikipedia data.

2) How Transformers harness multi-concept semantics for ICL.

Motivated by the aforementioned topical structure analysis of Li et al. [2023], and by empirical findings that Transformer representations obey a *linear geometric regularity* (high similarity within-concept, near-orthogonality across-concepts), Bu et al. [2025] studies in-context learning under a more realistic polysemous setting. Here data distribution associates each token with a sparse latent concept vector ($\boldsymbol{z}$), and each word ($\boldsymbol{x}$) and label ($\boldsymbol{y}$) has multiple concept-specific embedding components, allowing the same token to express different meanings depending on context. More formally, $\boldsymbol{z} \in \{0,1\}^K$ is the binary latent vector with $\boldsymbol{x} = \boldsymbol{M}\boldsymbol{z} + \boldsymbol{\xi}_x \in \mathbb{R}^{d_x}$ and $\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{z} + \boldsymbol{\xi}_y \in \mathbb{R}^{d_y}$, where $K < \min(d_x, d_y)$, $(\boldsymbol{\xi}_x, \boldsymbol{\xi}_y)$ is Gaussian distributed, and the columns of $\boldsymbol{M}$ and $\boldsymbol{Q}$ satisfy the linear geometric regularity conditions. Further assuming a contextual prior on $\boldsymbol{z}$ that ties in-context input samples $\begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_n & \boldsymbol{x}_{\text{query}} \\ \boldsymbol{y}_1 & \boldsymbol{y}_2 & \dots & \boldsymbol{y}_n & 0 \end{bmatrix}$ through a shared co-concept, this "multi-concept" model—which generalizes single-topic LDA sandboxes—captures contextual polysemy while preserving the structured geometry observed in pretrained models.

Under this setup, the authors analyze a two-layer Transformer (one attention layer followed by a ReLU feed-forward layer) trained on logistic loss when fed in-context classification prompts. They show that the model leverages the multi-concept geometry to learn the task efficiently, proving exponential convergence of the 0–1 error despite the non-linear attention-plus-MLP architecture. Further, they demonstrate that the Transformer still achieves Bayes-optimal classification error even under distribution shifts, such as new concept mixtures, altered concept frequencies, and previously unseen word–label pairings. Together, these results provide a theoretical account of how polysemous semantic geometry supports true out-of-distribution ICL, allowing Transformers to combine concepts in novel ways and solve unseen tasks.

3) How Transformers exhibit spontaneous topic changes.

Motivated by the contrast between abrupt, spontaneous topic shifts in human cognition and thought and the structured next-token prediction property of Transformers, Jia and Diaz-Rodriguez [2025] study when and why self-attention models exhibit spontaneous topic changes. To this end, they build on the Token-Priority Graph (TPG) framework of Li et al. [2024a], where each topic is defined as a set of TPGs. The key idea underpinning TPGs is that given a multi-class classification dataset of the form $\{(\boldsymbol{X}_i, y_i)\}_{i \in [N]}$ with the input sequence (sans the subscript i) $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_T]^\top \in \mathbb{R}^{T \times d}$ and next-token label $y \in [K]$, one can build a set of directed graphs, each indexed by one of the K possible last tokens $\boldsymbol{x}_T = \boldsymbol{e}_k$, containing directed edges $y \to x$ for each $x \in \boldsymbol{X}$. Here x denotes the discrete token corresponding to the token embedding $\boldsymbol{x} = \boldsymbol{e}_x$ and $\boldsymbol{e}_k$ is the embedding corresponding to the k^{th} token for $k \in [K]$. Fig. 12 illustrates this. The intuition here is that the edges capture the priorities across the tokens in a data sequence, conditioned on the last token of the input being $\boldsymbol{x}_T = \boldsymbol{e}_k$. Further, these graphs can be decomposed into strongly-connected components (SCCs), representing sets of tokens with equal priority, and induce a partial order across components. Thus, a "topic", a set of TPGs, can be thought of as a priority structure over tokens. Similar to how TPGs are generated from a dataset, equivalently, datasets can also be directly generated from the underlying TPGs, i.e. a topic.

Within this TPG framework, Li et al. [2024a] study the conditions under which a single-layer Transformer preserves or changes topics. They observe that models trained on mixed-topic corpora preserve the priority structure of the input topic: as long as higher-priority tokens of the current topic remain more frequent, the model stays anchored in that topic. A topic shift occurs only when tokens from another topic appear with sufficiently higher frequency to overturn the priority order. Moreover, longer sequences and ambiguous contexts stabilize the topic, reducing rather than increasing the chance of transition. Interestingly, this behavior contrasts sharply with human cognition, where longer discourse and ambiguity often encourage associative jumps.

More broadly, beyond the topic structure but still in the realm of NLP-inspired structured tasks, Yang et al. [2024] analyze the training dynamics of a one-layer Transformer on a basic word co-occurrence task. They show that, even from random initialization and without simplifying assumptions, gradient flow induces a two-phase learning process where an MLP first aligns with the signal and attention later sharpens the classification margin.

4.4 Multi-hop reasoning

Beyond their linguistic fluency, Transformers also achieve impressive performance on complex reasoning tasks, from multi-step factual reasoning to challenging math and coding problems [Lewkowycz et al., 2022]. A growing body of literature has analyzed such abilities from an expressivity power of view [Feng et al., 2023, Li et al., 2024b, Liu et al., 2023a, Sanford et al., 2024a], but our understanding of how such reasoning capabilities emerge during training remains nascent, particularly from a learning-dynamics perspective. Addressing this, Wang et al. [2025] provide one of the first theoretical accounts of how Transformers learn to solve *multi-step/multi-hop reasoning* tasks.

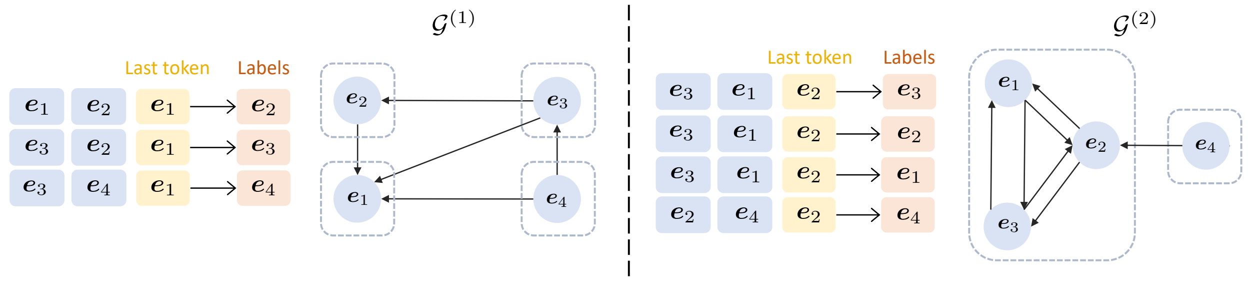

$\mathcal{G}^{(1)}$
Last token
Labels
e_1 e_2 e_1 → e_2
e_3 e_2 e_1 → e_3
e_3 e_4 e_1 → e_4
$\mathcal{G}^{(2)}$
e_3 e_1 e_2 → e_3
e_3 e_1 e_2 → e_2
e_3 e_1 e_2 → e_1
e_2 e_4 e_2 → e_4

Figure 12: **Illustration of token-priority graph (TPG).** Given the input sequences and labels (next tokens), we construct the TPGs $\{\mathcal{G}^{(k)}\}_{k=1}^{K}$ according to the last token. Two TPGs $\mathcal{G}^{(1)}$ (left) and $\mathcal{G}^{(2)}$ (right) are constructed using the samples with e_1 and e_2 as the last tokens, respectively. In each graph, directed edges (label token→input token) are added between tokens/nodes. Based on these directed edges, each graph can be partitioned into its strongly-connected components (SCCs, highlighted as dashed grey rectangles). Each SCC is a set of tokens where each token is reachable from every other token within that SCC. Adapted from Fig. 3 of Li et al. [2024a].

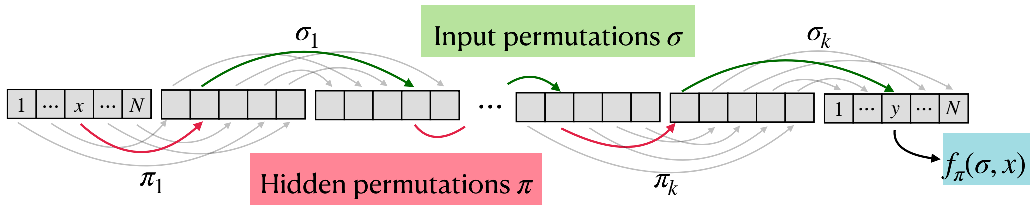


Figure 13: *k***-fold composition task**: red arrows represent the hidden permutation π_i and green arrows denote input permutations σ_i. Given an input $(\boldsymbol{\sigma}, x)$, $f_{\boldsymbol{\pi}}(\cdot,\cdot)$ composes $2k$ permutations to output $f_{\boldsymbol{\pi}}(\boldsymbol{\sigma}, x)$. Adapted from Fig. 1 of Wang et al. [2025].

As a motivating example, consider the classic two-hop reasoning task [Weston et al., 2014]: *John plays football. The football game is on Sunday. On what day does John play?* — where the model must compose the relations (John → Football, Football → Sunday) from context. A more complex variant, *John plays quarterback. The football game is on Sunday. On what day does John play?* requires combining contextual cues (John → Quarterback, Football → Sunday) with *global parametric knowledge* (Quarterback → Football), illustrating the broader challenge of multi-hop reasoning that integrates both in-context and stored relational knowledge.

Formalizing this, [Wang et al., 2025] introduce a *k-fold compositional* task, where the model must predict an element of $[N]$ by applying an interleaved sequence of k in-context permutations over $[N]$ and k hidden parametric permutations—effectively a k-hop reasoning problem that integrates contextual and latent parametric knowledge (Fig. 13). Concretely, let S_N be the set of permutations on N elements and let $\pi \circ \sigma$ denote the composition of two permutations $\pi, \sigma \in S_N$. Now let $\boldsymbol{\pi} = (\pi_1, \dots, \pi_k) \in (S_N)^k$ denote a sequence of k hidden permutations. The k-fold compositional task takes as input $(\boldsymbol{\sigma}, x) \in \mathcal{X} := (S_N)^k \times [N]$, where $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_k)$ represents a sequence of k in-context permutations and x an index in $[N]$. The target function $f_{\boldsymbol{\pi}} : \mathcal{X} \to [N]$ is then defined by

$$f_{\boldsymbol{\pi}}(\boldsymbol{\sigma}, x) = (\sigma_k \circ \pi_k \circ \sigma_{k-1} \circ \pi_{k-1} \circ \cdots \circ \sigma_1 \circ \pi_1)(x), \tag{2}$$

and the goal is to learn the function class $\mathcal{F} = \{f_{\boldsymbol{\pi}} : \boldsymbol{\pi} \in (S_N)^k\}$ under the uniform distribution on $\mathcal{X}$. In view of the earlier example, each contextual permutation σ_i encodes relations explicitly provided in the prompt (e.g. John → Quarterback), while each hidden permutation π_i captures implicit, parametric knowledge stored in the model weights (e.g., Quarterback → Football). Thus, the composed mapping $(\sigma_k \circ \pi_k \circ \cdots \circ \sigma_1 \circ \pi_1)(x)$ reflects a multi-hop reasoning chain that interleaves contextual and in-weights knowledge. For the Transformer architecture, they consider a L layer attention-only Transformer without causal masking and MLPs.

Towards analyzing the gradient dynamics, they first show that there exists a Transformer with depth $L = \log_2 k + 1$ and a suitable choice of embedding function and (key, query, value) such that it exactly solves the k-fold compositional task in Eq. (2) for all $(\boldsymbol{\sigma}, x) \in \mathcal{X}$. The main idea here is that of a recursive construction, where the first layer encodes the hidden permutation set $\boldsymbol{\pi}$ and ℓ^{th}-layer computes $\text{hop}_i^{2^{\ell-1}}(\boldsymbol{\sigma}, \cdot)$, where $\text{hop}_i^r(\boldsymbol{\sigma}, \cdot) = \sigma_{i+r-1} \circ \pi_{i+r-1} \circ \dots \sigma_{i+1} \circ \pi_{i+1} \circ \sigma_i \circ \pi_i$ denotes the interleaved composition from i to $i + r - 1$. By carefully choosing the model parameters, they show that a depth of $L = \log_2 k + 1$ suffices to represent $f_{\boldsymbol{\pi}}(\boldsymbol{\sigma}, x) = \text{hop}_1^k(\boldsymbol{\sigma}, x)$ for all $(\boldsymbol{\sigma}, x) \in \mathcal{X}$.

Motivated by this recursive representation and building upon the observation that learning difficulty grows with k while it's easier for a Transformer to learn from samples of $\text{hop}_i^r(\boldsymbol{\sigma}, \cdot)$, for much smaller $r \leq k$, they exploit the implied hierarchy to define a natural easy-to-hard curriculum. Concretely, at stage ℓ, the model is trained (gradient-based) on samples of $\text{hop}_i^{2^{\ell-1}}(\boldsymbol{\sigma}, \cdot)$, corresponding to compositions of length $2^{\ell-1}$, so that it progressively acquires competence on shorter compositions before advancing to longer ones. Capitalizing on this implicit curriculum, they prove that gradient-based training can recover $f_{\boldsymbol{\pi}}$ in $\text{poly}(Nk)$ samples and runtime, as opposed to a Statistical-Query (SQ) lower bound which requires exponential $N^{\Omega(k)}$ samples for the same task (with polynomial compute and access only to the target labels).

Along a similar vein, [Guo et al., 2025] emprically study the mechanisms underlying two-hop reasoning, one of the simplest instances of multi-hop tasks. They find that pretrained LLMs surprisingly fail at such tasks in the presence of distractors (information not relevant to the context), defaulting to random guessing. However, after minimal fine-tuning, models exhibit a sharp phase transition to near-perfect accuracy and strong generalization. Through reverse-engineering a three-layer Transformer trained from scratch, they uncover a clear progression in the attention dynamics: early layers retrieve intermediate entities, while later layers integrate them to produce the final inference, revealing a structured sequential query mechanism that captures the essence of reasoning within Transformer architectures.

5 Generalization

This section presents a brief discussion on generalization, and how understanding on representational and optimization results can help understand and improve generalization.

Failure modes due to attention and MLP The representability results in Section 3 reveal two failure modes in generalization, due to different components of the model architecture.

For MLP, which is used across all architectures, the challenge is rooted in the fact that it can only extrapolate linearly outside the range seen during training Xu et al. [2020]. This means that if a nonlinear function needs to be computed by MLP, then it will guarantee to be inaccurate outside the training range. One example is the task of parity (Section 3.1.1), where the MLP is intended to compute the mod function $(\sum_{i \in [T]} x_i) \mod 2$ and incurs errors empirically when $\sum_{i \in [T]} x_i$ changes (Liu et al. [2023a], Figure 13).

For attention, one failure mode is that the softmax may fail to be a good approximation for hardmax when used out of distribution (OOD). Recall from Section 3.1.1 that hardmax, or sparse attention, is used for many primitives such as adaptive selection. In practice, the hardmax is approximated by softmax, and the difference between the two increases when the softmax deminator increases, [10] which can happen when using a longer length, or when using same-length inputs but with a different token distributions Chiang and Cholak [2022], Liu et al. [2023b] In practice, a widely adopted mitigation is to adjust the softmax scaling when adjusting sequence length Chiang and Cholak [2022], which can be considered as a temperature parameter and is sometimes discussed in the context of positional encoding Su et al. [2021], Wang et al. [2024a].

Note that the above generalization failures are not fundamental to the change in length and arise for generical OOD scenarios even when the instance size is fixed.

Which tasks are more prone to generalization failure? Specific to length generaliztion, Zhou et al. [2024a] proposes a heuristic "RASP-generalization conjecture" that a task is likely length generalize when it can be represented with a short *RASP-L* program, a learnable variant of the RASP program Weiss et al. [2021]. While RASP is already designed with operations native to Transformers, RASP-L puts additional requirements on

[10] Specifically, suppose i^* is the position to be selected, i.e. we want its softmax value $\frac{\exp(\langle \boldsymbol{q}, \boldsymbol{k}_{i^*} \rangle)}{\exp(\langle \boldsymbol{q}, \boldsymbol{k}_{i^*} \rangle) + \sum_{j \in [T], j \neq i^*} \exp(\langle \boldsymbol{q}, \boldsymbol{k}_j \rangle)}$ close to one. However, since the exponential function is non-negative and bounded away from 0 when the parameter norms are bounded, the denominator grows as more tokens are included.

index operations. The RASP-generalization conjecture provides practically valuable guidelines. For instance, RASP-L makes predictions on whether a task is length generalizable (e.g. majority) versus not (e.g. parity), which generally align with empirical evidence. Moreover, modifying a task to make it RASP-L-compatible shows improved length generalization. Huang et al. [2024] later formalizes the conjecture using C-RASP Yang and Chiang [2024].

6 Discussion

This article surveyed results using sandboxes to provide more general insights. For representability (Section 3), we discussed tools for proving upper and lower bounds, as well as implications of these results on architectural comparisons and improvements. The optimiazation discussion (Section 4) focused on implicit bias of gradient-based methods on learning several sandbox tasks. Finally, for generalization (Section 5), we discuseed two failure modes due to MLP and attention respectively, and a promising framework for understanding length generalization.

While insights developed under simple sandboxes can be generally applicable, sandboxes can be limited since they are foundamentally simplications of the real world scenarios. Hence, an important future direction is to better understand the implications of simplifications and the interplay of different simplying assumptions. For data, most work assumes a single data source, though mixtures may be required for understanding certain phenomena such as the power law scaling. For models, some architectural details, such as tokenization and normalization Wortsman et al. [2023], Ahn et al. [2024], can have a surprisingly significant impact on optimization and warrant further study. It is also timely to find a proper sandbox for modeling pretrained models, which is useful for understanding emergent behaviors and post-training related properties.

References

K. Ahn, X. Cheng, M. Song, C. Yun, A. Jadbabaie, and S. Sra. Linear attention is (maybe) all you need (to understand transformer optimization). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=0uI5415ry7.

C. Anil, Y. Wu, A. Andreassen, A. Lewkowycz, V. Misra, V. Ramasesh, A. Slone, G. Gur-Ari, E. Dyer, and B. Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

S. Arora, A. Timalsina, A. Singhal, B. Spector, S. Eyuboglu, X. Zhao, A. Rao, A. Rudra, and C. Ré. Just read twice: closing the recall gap for recurrent language models. *arXiv preprint arXiv: 2407.05483*, 2024.

Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

P. Barceló, A. Kozachinskiy, A. W. Lin, and V. Podolskii. Logical languages accepted by transformer encoders with hard attention. *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2310.03817.

S. Bhattamishra, A. Patel, V. Kanade, and P. Blunsom. Simplicity Bias in Transformers and their Ability to Learn Sparse Boolean Functions. *Annual Meeting of the Association for Computational Linguistics*, 2022. doi: 10.48550/arXiv.2211.12316.

S. Bhattamishra, M. Hahn, P. Blunsom, and V. Kanade. Separations in the representational capabilities of transformers and recurrent architectures. *Advances in Neural Information Processing Systems*, 37:36002–36045, 2024.

A. Bietti, V. Cabannes, D. Bouchacourt, H. Jegou, and L. Bottou. Birth of a Transformer: A Memory Viewpoint. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A. Blakeman, A. Basant, A. Khattar, A. Renduchintala, A. Bercovich, A. Ficek, A. Bjorlin, A. Taghibakhshi, A. S. Deshmukh, A. S. Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022, 2003.

G. E. Blelloch. Prefix sums and their applications. 1990.

D. Bu, W. Huang, A. Han, A. Nitanda, T. Suzuki, Q. Zhang, and H.-S. Wong. Provable in-context vector arithmetic via retrieving task concepts. In *42nd International Conference on Machine Learning, ICML 2025*, 2025.

L. Chen, B. Peng, and H. Wu. Theoretical limitations of multi-layer transformer. *arXiv preprint arXiv: 2412.02975*, 2024a.

S. Chen, H. Sheen, T. Wang, and Z. Yang. Unveiling induction heads: Provable training dynamics and feature learning in transformers. *arXiv preprint arXiv:2409.10559*, 2024b.

T. Chen, T. Ma, and Z. Li. Non-asymptotic length generalization. *arXiv preprint arXiv:2506.03085*, 2025.

D. Chiang and P. A. Cholak. Overcoming a theoretical limitation of self-attention. *Annual Meeting of the Association for Computational Linguistics*, 2022. doi: 10.18653/v1/2022.acl-long.527.

D. Chiang, P. Cholak, and A. Pillay. Tighter bounds on the expressivity of transformer encoders. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR, 23-29 Jul 2023. URL https://proceedings.mlr.press/v202/chiang23a.html.

F. D'Angelo, F. Croce, and N. Flammarion. Selective induction heads: How transformers select causal structures in context. In *The Thirteenth International Conference on Learning Representations*, 2025.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

B. L. Edelman, E. Edelman, S. Goel, E. Malach, and N. Tsilivis. The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains, 2024.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. Das-Sarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

G. Feng, B. Zhang, Y. Gu, H. Ye, D. He, and L. Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.

P. Glorioso, Q. Anthony, Y. Tokpanov, J. Whittington, J. Pilault, A. Ibrahim, and B. Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv: 2405.16712*, 2024.

O. Golovneva, T. Wang, J. Weston, and S. Sukhbaatar. Contextual position encoding: Learning to count what's important. *arXiv preprint arXiv:2405.18719*, 2024.

N. Golowich, S. Jelassi, D. Brandfonbrener, S. M. Kakade, and E. Malach. The role of sparsity for length generalization in transformers. *arXiv preprint arXiv:2502.16792*, 2025.

A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv: 2312.00752*, 2023.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

T. Guo, D. Pai, Y. Bai, J. Jiao, M. I. Jordan, and S. Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. *arXiv preprint arXiv:2410.13835*, 2024.

T. Guo, H. Zhu, R. Zhang, J. Jiao, S. Mei, M. I. Jordan, and S. Russell. How do llms perform two-hop reasoning in context? *arXiv preprint arXiv:2502.13913*, 2025.

M. Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

Y. Hao, D. Angluin, and R. Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810, 2022. doi: 10.1162/tacl_a_00490. URL https://aclanthology.org/2022.tacl-1.46.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

X. Huang, A. Yang, S. Bhattamishra, Y. Sarrof, A. Krebs, H. Zhou, P. Nakkiran, and M. Hahn. A formal framework for understanding length generalization in transformers. *arXiv preprint arXiv: 2410.02140*, 2024.

S. Jelassi, S. d'Ascoli, C. Domingo-Enrich, Y. Wu, Y. Li, and F. Charton. Length generalization in arithmetic transformers. *arXiv preprint arXiv: 2306.15400*, 2023.

S. Jelassi, D. Brandfonbrener, S. M. Kakade, and E. Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

M. Jia and J. Diaz-Rodriguez. Dynamics of spontaneous topic changes in next token prediction with self-attention. *arXiv preprint arXiv:2501.06382*, 2025.

H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. SIAM, 2010.

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

E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, Cambridge, 1997.

N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching Arithmetic to Small Transformers. *arXiv preprint arXiv: 2307.03381*, 2023.

Y. Levine, N. Wies, O. Sharir, H. Bata, and A. Shashua. The depth-to-width interplay in self-attention. *NEURIPS*, 2020.

A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

B. Z. Li, Z. C. Guo, and J. Andreas. (how) do language models track state? *arXiv preprint arXiv: 2503.02854*, 2025.

Y. Li, Y. Li, and A. Risteski. How do transformers learn topic structure: towards a mechanistic understanding. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Y. Li, Y. Huang, M. E. Ildiz, A. S. Rawat, and S. Oymak. Mechanics of next token prediction with self-attention. In *International Conference on Artificial Intelligence and Statistics*, pages 685–693. PMLR, 2024a.

Z. Li, H. Liu, D. Zhou, and T. Ma. Chain of thought empowers transformers to solve inherently serial problems. *International Conference on Learning Representations*, 2024b. doi: 10.48550/arXiv.2402.12875.

B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Transformers Learn Shortcuts to Automata. 2023a. doi: 10.48550/arXiv.2210.10749. URL https://openreview.net/forum?id=De4FYqjFueZ.

B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Exposing Attention Glitches with Flip-Flop Language Modeling. *Neural Information Processing Systems*, 2023b.

C. Lu, Y. Schroecker, A. Gu, E. Parisotto, J. Foerster, S. Singh, and F. Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

A. V. Makkuva, M. Bondaschi, A. Girish, A. Nagle, H. Kim, M. Gastpar, and C. Ekbote. Local to Global: Learning Dynamics and Effect of Initialization for Transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

A. V. Makkuva, M. Bondaschi, A. Nagle, A. Girish, H. Kim, M. Jaggi, and M. Gastpar. Attention with Markov: A curious case of single-layer transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.

E. Malach. Auto-Regressive Next-Token Predictors are Universal Learners, 2023.

S. McLeish, A. Bansal, A. Stein, N. Jain, J. Kirchenbauer, B. R. Bartoldson, B. Kailkhura, A. Bhatele, J. Geiping, A. Schwarzschild, and T. Goldstein. Transformers can do arithmetic with the right embeddings. *arXiv preprint arXiv: 2405.17399*, 2024.

B. Merialdo. Tagging english text with a probabilistic model. *Computational linguistics*, 20(2):155–171, 1994.

W. Merrill and A. Sabharwal. The Expressive Power of Transformers with Chain of Thought. *arXiv preprint arXiv: 2310.07923*, 2023.

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

W. Merrill, A. Sabharwal, and N. A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

W. C. Merrill and A. Sabharwal. The Parallelism Tradeoff: Limitations of Log-Precision Transformers. *International Conference on Topology, Algebra and Categories in Logic*, 2022. doi: 10.1162/tacl_a_00562.

N. Nanda and T. Lieberum. A mechanistic interpretability analysis of grokking. *Alignment Forum*, 2022. URL https://www.alignmentforum.org/posts/N6WM6hs7RQMKDhYjB/a-mechanistic-interpretability-analysis-of-grokking.

E. Nichani, A. Damian, and J. D. Lee. How Transformers Learn Causal Structure with Gradient Descent. *arXiv preprint arXiv:2402.14735*, 2024.

J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

C. F. Park, E. S. Lubana, I. Pres, and H. Tanaka. Competition dynamics shape algorithmic phases of in-context learning. *arXiv preprint arXiv:2412.01003*, 2024.

B. Peng, S. Narayanan, and C. Papadimitriou. On limitations of the transformer architecture. *arXiv preprint arXiv: 2402.08164*, 2024.

J. Pérez, P. Barceló, and J. Marinkovic. Attention is Turing-Complete. *Journal of Machine Learning Research*, 22(75): 1–35, 2021.

L. Ren, Y. Liu, Y. Lu, Y. Shen, C. Liang, and W. Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv: 2406.07522*, 2024.

C. Sanford, D. J. Hsu, and M. Telgarsky. Representational strengths and limitations of transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 36677–36707. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/73bf692447f174984f30499ec9b20e04-Paper-Conference.pdf.

C. Sanford, D. Hsu, and M. Telgarsky. One-layer transformers fail to solve the induction heads task, 2024a. URL https://arxiv.org/abs/2408.14332.

C. Sanford, D. Hsu, and M. Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=QCZabhKQhB.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

R. Shen, S. Bubeck, R. Eldan, Y. T. Lee, Y. Li, and Y. Zhang. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023.

L. Strobl. Average-hard attention transformers are constant-depth uniform threshold circuits. *arXiv preprint arXiv:2308.03212*, 2023.

L. Strobl, W. Merrill, G. Weiss, D. Chiang, and D. Angluin. Transformers as Recognizers of Formal Languages: A Survey on Expressivity. *arXiv preprint arXiv: 2311.00208*, 2023.

L. Strobl, W. Merrill, G. Weiss, D. Chiang, and D. Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.

J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *NEUROCOMPUTING*, 2021. doi: 10.1016/j.neucom.2023.127063.

J. Team, B. Lenz, A. Arazi, A. Bergman, A. Manevich, B. Peleg, B. Aviram, C. Almagor, C. Fridman, D. Padnos, et al. Jamba-1.5: Hybrid transformer-mamba models at scale. *arXiv preprint arXiv:2408.12570*, 2024.

A. Varre, G. Yüce, and N. Flammarion. Learning in-context n-grams with transformers: Sub-n-grams are near-stationary points. *arXiv preprint arXiv:2508.12837*, 2025.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

R. Waleffe, W. Byeon, D. Riach, B. Norick, V. Korthikanti, T. Dao, A. Gu, A. Hatamizadeh, S. Singh, D. Narayanan, G. Kulshreshtha, V. Singh, J. Casper, J. Kautz, M. Shoeybi, and B. Catanzaro. An empirical study of mamba-based language models. *arXiv preprint arXiv: 2406.07887*, 2024.

J. Wang, T. Ji, Y. Wu, H. Yan, T. Gui, Q. Zhang, X. Huang, and X. Wang. Length generalization of causal transformers without position encoding. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14024–14040, Bangkok, Thailand, aug 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.834. URL https://aclanthology.org/2024.findings-acl.834.

J. Wang, D. Paliotta, A. May, A. M. Rush, and T. Dao. The mamba in the llama: Distilling and accelerating hybrid models. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/723933067ad315269b620bc0d2c05cba-Abstract-Conference.html.

M. Wang, R. Yu, L. Wu, et al. How transformers implement induction heads: Approximation and optimization analysis. *arXiv preprint arXiv:2410.11474*, 2024c.

Z. Wang, E. Nichani, A. Bietti, A. Damian, D. Hsu, J. D. Lee, and D. Wu. Learning compositional functions with transformers from easy-to-hard data. *arXiv preprint arXiv:2505.23683*, 2025.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

G. Weiss, Y. Goldberg, and E. Yahav. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090, 2021.

K. Wen, Y. Li, B. Liu, and A. Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded Dyck grammars. *Neural Information Processing Systems*, 2023.

K. Wen, X. Dang, and K. Lyu. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=h3wbI8Uk1Z.

J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

M. Wortsman, P. J. Liu, L. Xiao, K. Everett, A. Alemi, B. Adlam, J. D. Co-Reyes, I. Gur, A. Kumar, R. Novak, J. Pennington, J. N. Sohl-Dickstein, K. Xu, J. Lee, J. Gilmer, and S. Kornblith. Small-scale proxies for large-scale transformer training instabilities. *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2309.14322.

K. Xu, M. Zhang, J. Li, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

A. Yang and D. Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. *arXiv preprint arXiv: 2404.04393*, 2024.

A. Yang, M. Cadilhac, and D. Chiang. Knee-deep in c-rasp: A transformer depth hierarchy. *arXiv preprint arXiv:2506.16055*, 2025.

H. Yang, B. Kailkhura, Z. Wang, Y. Liang, et al. Training dynamics of transformers to recognize word co-occurrence via gradient flow analysis. *Advances in Neural Information Processing Systems*, 37:46047–46117, 2024.

S. Yao, B. Peng, C. Papadimitriou, and K. Narasimhan. Self-attention networks can process bounded hierarchical languages. *arXiv preprint arXiv:2105.11115*, 2021.

G. Yehudai, C. Sanford, M. Bechler-Speicher, O. Fischer, R. Gilad-Bachrach, and A. Globerson. Depth-width tradeoffs in algorithmic reasoning of graph tasks with transformers. *arXiv preprint arXiv: 2503.01805*, 2025.

C. Yun, S. Bhojanapalli, A. S. Rawat, S. Reddi, and S. Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

Y. Zhang, A. Backurs, S. Bubeck, R. Eldan, S. Gunasekar, and T. Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv: 2206.04301*, 2022.

H. Zhou, A. Bradley, E. Littwin, N. Razin, O. Saremi, J. M. Susskind, S. Bengio, and P. Nakkiran. What algorithms can transformers learn? A study in length generalization. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=AssIuHnmHX.

R. Zhou, C. Tian, and S. Diggavi. Transformers learn variable-order markov chains in-context. *arXiv preprint arXiv:2410.05493*, 2024b.

Y. Zhou, U. Alon, X. Chen, X. Wang, R. Agarwal, and D. Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024c.