# Compositions with Transformers

Luc Pommeret[1], Achraf Lassoued[2], and Michel de Rougemont[3]

[1]LISN and Paris-Cité University, France, `pommeret@lisn.fr`
[2]IRIF, France, `achraflassoued985@gmail.com`
[3]University Paris II and IRIF, France, `mdr@irif.fr`

November 20, 2025

**Abstract**

We study how Transformers may compose specific functions, contrary to the worst case scenario when the composition is error prone. We concentrate on games such as chess where runs in the PGN notation are given to a Transformer to predict the next token and the next move. The iteration of moves is a composition which is syntactically correct with high probability, although the Transformer ignores the syntactic rules of the game.

**Keywords:** Transformers, Function Composition, Neural Networks, Games, Chess

## Contents

## 1 Introduction

Transformers have become a cornerstone of modern artificial intelligence, demonstrating remarkable capabilities across a wide range of tasks. A key question in understanding these models concerns their ability to compose functions. That is, if a transformer can learn to compute functions $f$ and $g$ separately, how well can it compute their composition $f \circ g$ or simply $f \circ f$ or more generally $f^k$ ?

A Transformer takes $i \leq N$ consecutive tokens as input and defines a distribution $p$ for the $(i+1)$-th token. While theoretical worst-case analyses suggest that function composition in transformers should be error-prone [25], empirical evidence indicates that in many practical scenarios, transformers perform composition with surprising accuracy [14, 19].

We investigate this apparent contradiction by studying transformers in the context of games, which provide a controlled environment with well-defined rules and success metrics [19, 2]. Specifically, we focus on chess as our primary experimental domain following recent work [5, 14]. In chess, the transformer must learn to generate legal move sequences without explicit knowledge of the game's rules, effectively composing the relation of legal moves across multiple steps as demonstrated by recent studies on implicit world models [19, 28].

The Transformer is given samples from a distribution $p$ and learns a distribution $p_\Theta$ close to $p$ where $\Theta$ is the set of parameters defining the Attention mechanism and the Perceptron model. We examine the distribution $p_\Theta$ of moves predicted by the model given a partial run taken from $p$ and analyze the consistency of the moves, i.e. the probability that the move follows the Chess rules. The consistency with game rules throughout extended sequences follows the methods proposed in [5, 28].

The main results of our work demonstrate that:

- If the KL divergence between $p$ and $p_\Theta$ is small, then the consistency of the moves can be guaranteed in expectation, and the error rate in function composition does not accumulate significantly during sequential prediction tasks.
- Empirical analysis supports theoretical bounds on the likelihood of illegal moves in sequential predictions

These findings contribute to our understanding of transformer capabilities and limitations, particularly in tasks requiring compositional reasoning.

### 1.1 Comparison with other approaches

In his seminal work, [10] studied the *Identification of a Language in the Limit.* Assume some infinite public enumeration of infinite languages $\mathcal{L} = (L_1, L_2, ...L_i, .......)$ with possible repetitions and a game between an adversary $\mathcal{A}$ and a learner $\mathcal{B}$: the adversary selects an index $i*$ and generate at each step $j = 1, 2...$ a new word $w \in L_{i*}$. The Learner outputs at each stage $j$ some index $i_j$ and wins if $L_{i_j} = L_{i*}$. The Learner *Identifies the family L in the limit* if there is some $j_0$ such that the Learner always wins after stage $j_0$, i.e. $\forall j > j_0 \;\; L_{i_j} = L_{i*}$.

In some recent work, [16] studied a similar notion, *Generation of a Language in the Limit*, where the Learner only needs to generate a new word of $L_{i*}$. If $S_j = \{w_1, .....w_j\}$ is the set of words generated by $\mathcal{A}$ in some $L_{i*}$ at stage $j$, the Learner wins if he generates a word $w' \in L_{i*} - S_j$ at stage $j$. Similarly, the Learner *Generates a language of L in the limit* if there is some $j_0$ such that the Learner always wins after stage $j_0$, i.e. $\forall j > j_0 \;\; L_{i_j} = L_{i*}$.

In [13], the authors show some Trade-Offs between the error probability of a generator and the possible mode collapse, i.e. the generation of a fraction of the hidden language $L_{i*}$. In our setting, we don't have an enumeration $\mathcal{L}$ of languages $L_i$ but a large finite language of valid PGN sequences given from a distribution

$p$ and we learn a distribution $p_\Theta$. If we assume that the $KL$ divergence between these two distributions is small, we will prove that the error probability is small.

## 1.2 Main definitions

Most of Compexity theory concentrates on the worst-case scenario. We introduce a different notion of computation *relative to a distribution* $p$ on the inputs in order to precisely define the correction of a computation. We only guarantee a correct output if the input follow $p(x_1, ...x_i)$ for $i \leq N$. It generalizes the notion of *one-sided* or *two-sided stochastic randomized* algorithm $A$ for a language $L$ introduced in **??** for inputs which follow a distribution $\mu$. In that setting, there are two different independent probabilistic spaces, $\Omega$ the internal random choices of the algorithm and the probabilistic distribution $\mu$ of the input. For a two-sided stochastic approximation, we required that:

- If $x \in L$ is drawn from $\mu$, $Prob_{\mu\times\Omega}[A(x)\ accepts] \geq 1 - \delta$
- If $x \notin L$ is drawn from $\mu$, $Prob_{\mu\times\Omega}[A(x)\ rejects] \geq 1 - \delta$

For the one-sided version, we rely on the worst-case when $x \in L$, i.e.
• If $x \in L$, $Prob_\Omega[A(x)\ accepts] \geq 1 - \delta$

In our case a Transformer computes $A(x_1, ...x_i) = x_{i+1}$, the next token choosen with probability $p_\Theta(x_{i+1} \mid x_1, ...x_i)$. How do we define whether the output of a Transformer $p, p_\Theta$ satisfies a property $P$ ? We consider inputs $x_1, ...x_i$ taken from $p$ and $x_{i+1}$ is taken from $p_\Theta$. The two distributions are not independent.

**Definition 1** *A Transformer* $(p, p_\Theta)$ *output satisfies a property* $P$ *relative to* $p$ *if for all* $0 \leq i \leq N$*:*

$$Prob_{p\times p_\Theta}[A(x_1, ...x_i) = x_{i+1} \in P] \geq 1 - \delta$$

As an example $P$ is the property that $x_{i+1}$ is a valid Chess move taken by the Transformer with the distribution $p_\Theta$, given a prefix $x_1, ...x_i$ of valid moves taken from $p$. We can also write:

$$\mathbb{E}_{p(x_1,...x_i)}\ Prob_{p_\Theta}[A(x_1, ...x_i) = x_{i+1} \in P] \geq 1 - \delta$$

**Probing.** The notion of probing was introduced in [11] to compute some property of the input $x$, reading some internal layers of the Transformer instead of the original input $x$. Given some internal layers of a Transformer, a probe constructs a new distribution $q_\Theta$ with new layers, based on samples from $p$. We view the property of the input as a Dirac distribution $q$ and we want that the KL divergence between $q$ and $q_\Theta$ to be small.

Notice that in this model, the probe is a randomized algorithm as the output is a distribution. In the Chess example, a Transformer predicts the next moves. A probe read some internal layers, add new layers to can predict the board positions [], i.e. which Chess pieces are in position $(i, j)$ at a given stage. In general it is a heatmap with some uncertainty, represented by a distribution $q_\Theta$.

Let $Z$ be a subset of the layers of the multilayers Transformer, taking $x$ as input. Assume we want to compute a new function $f(x)$, taking some internal values $z$ from $Z$ as input. We train new layers based on samples $\{z_j, f(x_j)\}_j$ instead of the samples $\{x_j, f(x_j)\}_j$. For simplicity assume the range of $f$ is $\{0, 1\}$. A *linear probe* computes $g(z)$, as $z$ is computed from $x$, by a linear transformation $\sum w.z + b$. Assume that the result is a distribution $q_{\Theta'}$ over $\{0, 1\}$ where $\Theta'$ is the list of new parameters. A general probe may also have several layers as a new Transformer.

**Definition 2** *A discrete function $f(x)$ where $x = x_1, \ldots x_i$ can be $\delta$-probed from a Transformer if there are some layers values $z$ of the Transformer and new layers defining $g(z)$ such that:*

$$Prob_{p(x)\times q_{\Theta'}}\ [f(x) = g(z)] \geq 1 - \delta$$

*A function $f(x)$ whose range is a numerical value can be $(\varepsilon, \delta)$-probed from a Transformer if there are some layers values $z$ such that for all $x$:*

$$Prob_{p(x)\times q_{\Theta'}}\ [f(x) - g(z)| \leq \varepsilon] \geq 1 - \delta$$

**Compositions.** The Transformer defines a distribution $p_\Theta$. On an input $x = x_1, x_2 \ldots x_i$ it defines a random variable $x_{i+1}$ with probability $p_\Theta(x_{i+1} \mid x_1, \ldots x_i)$. Let $Next(x_1, x_2 \ldots x_i) = x_{i+1}$ and $Next(x_1, x_2 \ldots x_i, x_{i+1}) = x_{i+2} = Next^2(x_1, x_2 \ldots x_i)$. More generally $x_{i+k} = Next^k(x_1, x_2 \ldots x_i)$.

We assume that there is an easy to compute function $h$ such that $z = h \circ Next^k(x_1, x_2 \ldots x_i)$. It takes the output of the Transformer and produces the result. Given two functions $\pi_1$ and $\pi_2$, with the same domain and range $D$ we define their possible composition as follows:

**Definition 3** *Given two functions $\pi_1$ and $\pi_2$ and a Transformer, let $z = h \circ Next^k(x)$ where $Next$ is the generative function of the Transformer. The two functions $\delta$-compose if there exists some function $h$ and some integer $k$ such that:*

$$Prob_{p(x)\times p_\Theta}\ [z = \pi_2 \circ \pi_1(x))] \geq 1 - \delta$$

As an example, suppose we have a domain $D$ with individuals $\{John, Mary, \ldots\}$, Cities $\{London, L.A., S.F., \ldots\}$ and Countries $\{U.S., U.K, \ldots\}$ and two binary relations. The first relation *Live*$(x, y)$ states that individual $x$ lives in City $y$ (*John lives in London*) and the second relation *Location*$(y, z)$ states that City $y$ is located in the Country $z$ (*London is in the U.K.*). We then ask: *In which Country does John live?*. In this case $y = \pi_1(x)$ gives the City $y$ such that *Live*$(x, y)$ and $z = \pi_2(y)$ is the Country such that *Location*$(y, z)$.

## 2 Transformers

We first define a precise model for a Transformer and then consider the world of Chess where runs are given in the PGN format with a small number of tokens. The Transformer is given many runs taken with a distribution $p$ and learns a distribution $p_\Theta$ of $N + 1$ tokens. It can then predict the next token with a probability $p_\Theta(x_{i+1} \mid x_1, \ldots x_i)$ for $i = 0, \ldots N$. We study the composition of Chess moves in this context.

A transformer model is characterized by several key parameters:

- $d$ - dimension of the embedding vectors
- $n$ - number of tokens in the vocabulary
- $N$ - maximum sequence length the model can process

Additional parameters that define the architecture include:

- $H$ - number of attention heads
- $D$ - dimension of the feed-forward network's hidden layer
- $L$ - number of transformer layers
- $p$ - numerical precision of the weights

A *decoder-only* Transformer takes $N$ consecutive tokens as input and produces a probability distribution over the vocabulary for the $(N + 1)$-th token. The architecture consists of two main components: the self-attention mechanism and the feed-forward neural network.

## 2.1 Self-Attention Mechanism

For simplicity, let's first consider a single attention head ($H = 1$). The mechanism involves three learned matrices:

- $Q \in \mathbb{R}^{d \times d}$ - query transformation matrix
- $K \in \mathbb{R}^{d \times d}$ - key transformation matrix
- $V \in \mathbb{R}^{d \times d}$ - value transformation matrix

Consider a sequence of $N$ tokens: $t_1, \ldots, t_N$ where $x_i \in \mathbb{R}^d$ is the embedding of token $t_i$. The computation proceeds as follows:

1. First, positional encoding is added to each embedding: $x_i' = x_i + \mathrm{pos}(i)$ where $\mathrm{pos}(i)$ is the positional encoding for position $i$
2. For each position $i$, compute query, key, and value vectors:
$$q_i = (Q \cdot x_i')^T, \quad k_j = K \cdot x_j', \quad v_j = V \cdot x_j'$$
3. Calculate attention weights using scaled dot-product (for $j = 1, \ldots N$) :
$$(r_{i,1}, \ldots, r_{i,j}) = \mathrm{Softmax}\left(\frac{q_i \cdot k_1}{\sqrt{d}}, \ldots, \frac{q_i \cdot k_j}{\sqrt{d}}\right)$$
4. Compute the weighted sum of values:
$$y_i = \sum_{j \le i} r_{i,j} \cdot v_j \tag{1}$$
5. Apply residual connection and layer normalization:
$$y_i' = \mathrm{LayerNorm}(x_i' + y_i)$$

This process transforms the sequence of embeddings $x_1, \ldots, x_N$ into new vectors $y_1', \ldots, y_N'$ of the same dimension $d$. The weighted sum (1) is the *decoder* version of a Transformer as opposed to the *encoder* version where the sum would be $y_i = \sum_{j=1}^{N} r_{i,j} \cdot v_j$.

## 2.2 MLP: Multi-Layer Perceptron

After the self-attention mechanism, each position passes through a feed-forward neural network with 2 layers.

1. Apply a two-layer perceptron with ReLU activation:
$$z_i' = W_2 \cdot \mathrm{ReLU}(W_1 \cdot y_i')$$
where $W_1 \in \mathbb{R}^{D \times d}$ and $W_2 \in \mathbb{R}^{d \times D}$
2. Apply residual connection and layer normalization:
$$z_i = \mathrm{LayerNorm}(y_i' + z_i')$$

The function $\mathrm{ReLU}(x) = \max(0, x)$ is applied element-wise.

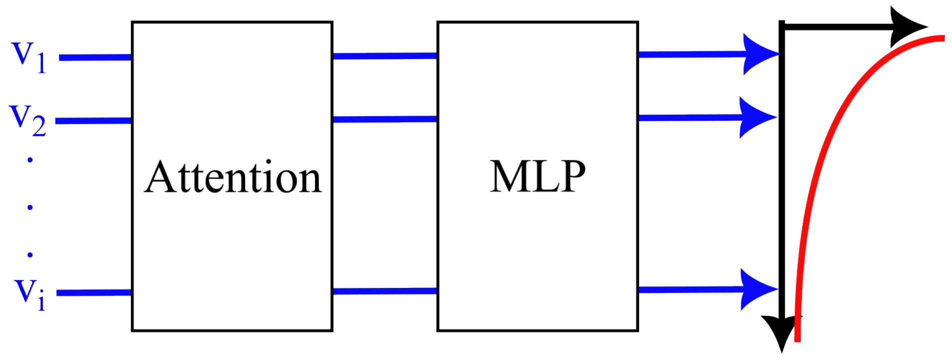


Figure 1: Transformer Architecture of one layer.

## 2.3 Output Distribution

After processing through all layers, the final output is transformed into a probability distribution over the vocabulary. For $i = 1, \ldots N$:

$$D_{\text{output}}(x_1, \ldots, x_i) = \text{Softmax}(W_0 \cdot z_i)$$

where $W_0 \in \mathbb{R}^{n \times d}$ is a learned parameter matrix and $z_i$ is the embedding of the last token after all transformations.

The complete transformer model is thus parameterized by the matrices $Q, K, V \in \mathbb{R}^{d \times d}$ (for each head and layer), $W_0 \in \mathbb{R}^{n \times d}$, $W_1 \in \mathbb{R}^{D \times d}$, $W_2 \in \mathbb{R}^{d \times D}$ (for each layer), the embeddings for each token and the parameters of the LayerNorm operations.

## 2.4 Multiple Heads and Layers

The transformation $x_i \to z_i$ constitutes one layer. In practice, transformers use multiple attention heads and stack multiple layers:

- For $H > 1$ heads, the embedding is partitioned into $H$ vectors of dimension $d/H$, with each head having its own set of matrices $Q, K, V \in \mathbb{R}^{d/H \times d/H}$
- The outputs from all heads are concatenated and linearly transformed
- The entire process is repeated $L$ times with different parameters for each layer

## 2.5 Transformer as a Generator

A key property of transformer models is their ability to generate sequences by iteratively predicting the next token. Given a sequence $x_1, \ldots, x_i$ with $i \leq N$, the model selects $x_{i+1}$ according to the conditional distribution $p(x_{i+1} \mid x_1, \ldots, x_i)$.

The autoregressive generation process works as follows:

- Starting with an empty sequence, we sample $x_1$ with probability $p(x_1)$
- We then sample $x_2$ with probability $p(x_2 \mid x_1)$
- This continues until we have generated the desired sequence length

This process implicitly defines a joint probability distribution over sequences:

$$p(x_1, \ldots, x_{N+1}) = p(x_1) \cdot p(x_2 \mid x_1) \cdot \ldots \cdot p(x_{N+1} \mid x_1, \ldots, x_N)$$

An important observation is that this represents a highly efficient compression of the full joint distribution. The support of the complete distribution is of size $n^{N+1}$, which would require exponential space to represent explicitly. However, the transformer parameterizes this distribution using only:

$$O(L \cdot (d^2 + n \cdot d) + N \cdot d)$$

parameters.

This efficient compression allows transformers to model complex sequence distributions with a manageable number of parameters, which is crucial for their practical application in language modeling and related tasks [24, 11, 30].

### 2.6 Probing new properties

We concentrated our analysis not only on the legality of moves defined by the Transformer but also on other properties such as the prediction of chess pieces in position $(i, j)$ of the board and more generally of the global board prediction. We also considered the ELO rating associated with a game.

Can the internal states of the Transformer be used for such tasks? The notion of probing, introduced in [11] and further developed in [4, 1, 12], uses the concept of a computation *relative to a Neural Network* and more generally *relative to an Algorithm.* [8, 7] study how to obtain syntactic information by probing.

The board prediction is a discrete function whereas the ELO rating can be viewed as a continuous function.

## 3 Chess as a Compositional Benchmark

We focus on chess as our primary experimental domain to investigate the composition capabilities of transformer models. Chess serves as an ideal benchmark for several reasons:

- It has well-defined, deterministic rules
- It offers a clear success metric (legal vs. illegal moves)
- It provides a standardized rating system (Elo) that allows for precise performance measurement [31]
- It requires compositional reasoning across multiple moves
- It has a rich history of documented games for training data [20]

### 3.1 Chess as a Language Modeling Task

Rather than approaching chess as a strategic game requiring search and evaluation, we frame it purely as a language modeling task. We train models on chess games written in Portable Game Notation (PGN) format, which consists of sequences of moves using standard algebraic notation:

```
1.e4 e5 2.Nf3 Nc6 3.Bb5 a6 4.Ba4 Nf6 ...
```

In this formulation, the model must predict the next token given the history of previous tokens, without any explicit encoding of the board state or the rules of chess. This differs fundamentally from traditional chess engines, which rely on sophisticated position evaluation functions and extensive search algorithms.

To analyze the compositional aspects of chess prediction, we formalize chess positions and moves in terms of relations rather than functions. This approach better captures the one-to-many nature of chess moves, as multiple legal moves are possible from any given position.

Let $\{P, R, N, B, Q, K\}$ represent the chess pieces (Pawn, Rook, Knight, Bishop, Queen, and King). We define a board position as a function:

$$\begin{aligned} bp : \{a, b, \ldots, h\} \times \{1, 2, \ldots, 8\} \rightarrow \\ (\{P, R, N, B, Q, K\} \times \{White, Black\}) \cup \{\emptyset\} \end{aligned} \tag{2}$$

This function assigns either a piece of a specific color or nothing to each square on the board. For any coordinates $(i, j)$, $bp(i, j) = (p, c)$ indicates that a piece of type $p$ with color $c$ is on square $(i, j)$, while $bp(i, j) = \emptyset$ indicates that the square is empty.

## 3.2 From Token-Level to Move-Level Distribution

The transformer model defines a distribution $p'_\Theta(t_N \mid t_1, \ldots, t_{N-1})$ over the next token $t_N$ given the previous tokens $t_1, \ldots, t_{N-1}$, where $\Theta$ is the set of parameters defining the Transformer.

However, a complete chess move typically consists of multiple tokens (e.g., "Nf3" consists of tokens 'N' and 'f3', or might be tokenized differently depending on the tokenization scheme). A move consist of a blank token after the first token, and a blank token after the second token. For example, "Nf3" might be tokenized as "N", "f3", and a blank token. We can derive a move-level distribution from the token-level distribution:

$$p_\Theta(m_k \mid m_1, \ldots, m_{k-1})$$

where $m_i$ represents a complete chess move. This move-level distribution is what we analyze in order to understand the model's ability to maintain legal play across multiple moves. Let $s_{k-1}$ be the state of the board after the moves $m_1, \ldots, m_{k-1}$

$$p_\Theta(m_k \text{ is legal}) = \sum_{m \in T(s_{k-1}, m_k)} p(m_k \mid m_1, \ldots, m_{k-1})$$

where $T(s_{k-1}, m_k)$ if $m_k$ is a legal move from state $s$.

By analyzing these transition probabilities across different positions and game phases, we can assess the model's understanding of chess rules and its ability to maintain legal play through composition.

## 3.3 Compositional Aspects of Chess

This relational formulation highlights the compositional nature of chess gameplay:

- A complete game can be viewed as a path in a directed graph where vertices are board positions and edges are labeled by moves $m_k$ such that $T(s_{k-1}, m_k)$. An edge connects a state $s_{k-1}$ with a state $s_k$, the result of the move $m_k$.
- The composition of $T$ with itself $k$ times, denoted $T^k$, represents positions reachable in exactly $k$ moves

- Learning to play chess involves learning to compose the relation $T$ repeatedly while maintaining legal play

A central question in our investigation is how errors accumulate during sequential predictions. Remarkably, we observe that for well-trained transformer models, the error rate does not grow significantly with the number of moves, despite the exponential growth in the space of possible positions.

This suggests that transformers can effectively learn to compose the legal move relation across multiple steps, maintaining consistency with the rules even though they were never explicitly programmed with them. This compositional capability emerges purely from exposure to examples of legal chess games during training.

### 3.4 Distribution on Chess Moves

While transformers operate at the token level, chess gameplay occurs at the level of moves. This distinction is important for understanding how the model composes legal play sequences.

When generating chess moves, two types of errors can occur:

- **Syntactic errors**: The generated token sequence does not form a valid move in algebraic notation (e.g., "NNf7" or "e9")
- **Semantic errors**: The generated move follows correct algebraic notation but is illegal in the current board position (e.g., moving a piece to a square occupied by another piece of the same color)

Our analysis focuses on both types of errors, but particularly on semantic errors, which reveal whether the model has learned to compose legal move sequences rather than just the syntax of chess notation.

## 4 Composition

We first consider the worst-case point of view, using Communication Complexity and then the composition under a distribution $p$ as defined in the intoduction.

### 4.1 Communication Complexity of Functional Composition

We sketch the worst-case analysis from [25, 6] for the composition capabilities of Transformers. We use Communication Complexity [18], to study functional composition: given functions $f : A \to B$ and $g : B \to C$, we want to compute their composition $f \circ g : A \to C$, where $(f \circ g)(x) = f(g(x))$. With the Communication Complexity lens, we can formulate the following scenario: Alice has a function $f$, Bob has computed $i = g(x)$ for some input $x$ given to both Alice and Bob, and they need to collaborate to compute $f(i)$. The *One way Communication Complexity* measures the number of bits exchanged from Alice to Bob in the worst-case. This is essentially the Index function problem in Communication Complexity.

Consider sets $A$,$B$,$C$, each of size $n$. We have functions $f : B \to C$ and $g : A \to B$, and we want to compute $f(g(x))$ for some $x \in A$.

In our analysis, we introduce three key random variables:

- $i^* = g(x)$ - the intermediate result
- $\Pi$ - the message exchanged from Alice to Bob (or the internal representation in a transformer)
- $f(i^*)$ - the final output

In the appendix, we prove [25]'s main lemma 9.

If $\mathcal{T}$ be a transformer with parameters $H$ (number of heads), $d$ (embedding dimension), and $p$ (precision) and $|\Pi| = n \log n - R$, then the probability over random functions $f, g$ and input $x$ that $\mathcal{T}$ computes $f(g(x))$ incorrectly is at least $\frac{R}{3n \log n}$.

By definition of the conditional entropy and mutual information:

$$H[f(i^*) \mid \Pi, i^*] = H[f(i^*) \mid i^*] - I[\Pi; f(i^*) \mid i^*]$$

Fano's inequality relates the error probability $\delta = \mathbb{P}[error]$ to the conditional entropy:

$$H[f(i^*) \mid \Pi, i^*] \leq H(error) + \delta \cdot \log n$$

where $H(error) = -\delta \log \delta - (1-\delta) \log(1-\delta)$ is the binary entropy function. It implies that $\delta ::_g eq \frac{R}{3n \log n}$. This lemma provides a fundamental limitation: in the worst case over random functions, a transformer model with limited capacity will fail to compute compositions correctly with high probability.

### 4.1.1 Auto regressive Communication

## 4.2 Composition under a distribution

The worst-case analysis assumes uniform distributions over functions and inputs. However, in real-world applications, the distributions are rarely uniform. We now analyze how biased distributions affect composition performance. In many practical scenarios, the intermediate values $i^* = g(x)$ follow biased distributions, such as power-law distributions common in natural data.

### 4.2.1 Entropy

**Lemma 1 (Entropy of Power-Law Distribution)** *For a power-law distribution $p(i) = \frac{c}{i^\alpha}$ where $\alpha > 1$ and $c$ is a normalization constant, the entropy $H(p)$ is finite and independent of the size of the domain.*

**Proof.** The entropy of $p$ is:

$$\begin{aligned} H(p) &= -\sum_i p(i) \log p(i) = -\sum_i \frac{c}{i^\alpha} \log\left(\frac{c}{i^\alpha}\right) \\ &= -\sum_i \frac{c}{i^\alpha}(\log c - \alpha \log i) = -\log c \sum_i \frac{c}{i^\alpha} + c.\alpha \sum_i \frac{\log i}{i^\alpha} = -\log c + c.\zeta'(\alpha) \end{aligned}$$

The first sum equals 1 by normalization, and the second sum is the derivative of the Riemann's $\zeta(\alpha)$ function which converges to a constant for $\alpha > 1$. Thus, $H(p)$ is finite and independent of the domain size. □

Under biased distributions, the success probability for composition can be much higher than in the worst case. Consider the basic scenario where the distribution on $x$ implies a distribution on $i = g(x)$ which is a power law.

**Lemma 2 (CC Under Biased Distribution)** *The Communication Complexity if the composition under a biased distribution is constant*

**Proof.**

Alice sends the most frequent values to Bob, low error.

□

When the representation capacity of the transformer is sufficient to capture the entropy $H(p)$ of the intermediate distribution, the mutual information $I[\Pi; f(i^*) \mid i^*]$ approaches $H[f(i^*) \mid i^*]$, making the conditional entropy $H[f(i^*) \mid \Pi, i^*]$ close to zero. This implies that $\delta$ must be small for the overall entropy to be low, leading to a high probability of successful composition.

Let $\mathcal{T}$ be a transformer with parameters such that the capacity of its internal representation accommodates the entropy $H(i)$ of the distribution of intermediate values. Then the probability of successful composition is at least $1 - \delta$, where $\delta$ depends on the relationship between the model capacity and $H(i)$.

This result explains why transformers can perform well on composition tasks in practice, despite the theoretical worst-case limitations. When the distribution of intermediate values has low entropy (as is often the case in natural data), the model can effectively learn to compose functions with high accuracy.

#### 4.2.2 Analysis based on the $\mathsf{KL}$ divergence

We now develop a formal analysis of the probability that a transformer model composes functions with high probability, under the distribution $p_\Theta$ learnt by the Transformer.

We apply this approach to understand the probability to observe illegal moves when the Transformer generates new moves, given a valid prefix. We demonstrate that, under reasonable assumptions about model training, the probability of generating illegal moves remains low even across extended move sequences.

Let $p(x_1, \ldots, x_m)$ represent the true distribution of legal chess games from a dataset such as Lichess, and let $p_\Theta(x_1, \ldots, x_m)$ be the distribution learned by a transformer with parameters $\Theta$. Here, each $x_i$ represents a complete move pair (one move by White followed by one move by Black).

We assume that the Transformer has been trained to approximate the true distribution, such that the Kullback-Leibler divergence between the two distributions is small:

$$\mathsf{KL}(p \parallel p_\Theta) \leq \varepsilon$$

for some small $\varepsilon > 0$.

#### 4.2.3 Properties of Marginal Distributions

Our analysis proceeds along these steps:

1. We show that if the KL divergence between the joint distributions is small, then the KL divergence between the marginals on the first dimension is also small,

2. For the first conditional probabilities $p(x_2|x_1)$, and $p_\Theta(x_2|x_1)$ defined by the Transformer, we show that the expectation over $p(x_1)$ of the KL divergence is also small. Similarly for $p(x_{j+1}|x_1, x_2 \ldots x_j)$, the expectation over $p(x_1, x_2 \ldots x_j)$ of the KL divergence is also small.

3. Using Pinsker's inequality, a small KL divergence implies a small $L_1$ distance between distributions. A small $L_1$ distance implies a low probability of generating illegal moves

We begin with the relationship between joint and marginal KL divergences:

**Lemma 3 (KL Divergence of Marginals)** *If* $\mathsf{KL}(p(x_1, x_2) \parallel q(x_1, x_2)) \leq \varepsilon$, *then* $\mathsf{KL}(p(x_1) \parallel q(x_1)) \leq \varepsilon$ *and* $\mathbb{E}_{p(x_1)} \mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \leq \varepsilon$.

**Proof.** By definition of the KL divergence:

$$\begin{aligned}
\mathsf{KL}(p(x_1,x_2) \parallel q(x_1,x_2)) &= \sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_1,x_2)}{q(x_1,x_2)}\right) \\
&= \sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_1)p(x_2|x_1)}{q(x_1)q(x_2|x_1)}\right) \\
&= \sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_1)}{q(x_1)}\right) \\
&+ \sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_2|x_1)}{q(x_2|x_1)}\right)
\end{aligned}$$

The first term can be simplified:

$$\begin{aligned}
\sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_1)}{q(x_1)}\right) &= \sum_{x_1}\left(\sum_{x_2} p(x_1,x_2)\right) \log\left(\frac{p(x_1)}{q(x_1)}\right) \\
&= \sum_{x_1} p(x_1) \log\left(\frac{p(x_1)}{q(x_1)}\right) \\
&= \mathsf{KL}(p(x_1) \parallel q(x_1))
\end{aligned}$$

For the second term, we write:

$$\begin{aligned}
\sum_{x_1,x_2} p(x_1,x_2) \log\left(\frac{p(x_2|x_1)}{q(x_2|x_1)}\right) &= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2|x_1) \log\left(\frac{p(x_2|x_1)}{q(x_2|x_1)}\right) \\
&= \sum_{x_1} p(x_1) \cdot \mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1))
\end{aligned}$$

Since $\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \geq 0$ for any distributions, the second term is non-negative. Therefore:

$$\begin{aligned}
\mathsf{KL}(p(x_1,x_2) \parallel q(x_1,x_2)) &= \mathsf{KL}(p(x_1) \parallel q(x_1)) \\
&+ \sum_{x_1} p(x_1) \cdot \mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \\
&= \mathsf{KL}(p(x_1) \parallel q(x_1)) + \mathbb{E}_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \qquad (3) \\
&\geq \mathsf{KL}(p(x_1) \parallel q(x_1)) \\
&\geq \mathbb{E}_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1))
\end{aligned}$$

Since $\mathsf{KL}(p(x_1,x_2) \parallel q(x_1,x_2)) \leq \varepsilon$ by assumption, we conclude that $\mathsf{KL}(p(x_1) \parallel q(x_1)) \leq \varepsilon$ and $\mathbb{E}_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \leq \varepsilon$. □

Notice that we can also infer that $p(x_1) \cdot \mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1))$ is small. However, $p(x_1)$ can be very small and we can't directly infer that $\mathsf{KL}(p(x_2|x_1 \parallel q(x_2|x_1))$ is also small. We can only infer that the expectation over $p(x_1)$ of $\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1))$ is small. We generalize this argument for $p(x_1,x_2,x_3)$ and then formally for $p(x_1,x_2,...x_m)$. By writing $p(x_2,x_3|x_1) = p(x_2|x_1).p(x_3|x_1,x_2)$ and similarly for $q$, we apply lemma 3 and write:

$$\begin{aligned}
\mathsf{KL}(p(x_2,x_3|x_1) \parallel q(x_2,x_3|x_1)) &= \mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \\
&+ \sum_{x_2} p(x_2|x_1) \cdot \mathsf{KL}(p(x_3|x_1,x_2) \parallel q(x_3|x_1,x_2)) \\
E_{p(x_1)}\mathsf{KL}(p(x_2,x_3|x_1) \parallel q(x_2,x_3|x_1)) &= E_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \\
&+ \sum_{x_1} p(x_1) \sum_{x_2} p(x_2|x_1) \cdot \mathsf{KL}(p(x_3|x_1,x_2) \parallel q(x_3|x_1,x_2))
\end{aligned}$$

The last term is:

$$\mathbb{E}_{p(x_1,x_2)}\mathsf{KL}(p(x_3|x_1,x_2) \parallel q(x_3|x_1,x_2))$$

Therefore

$$\mathbb{E}_{p(x_1)}\mathsf{KL}(p(x_2,x_3|x_1) \parallel q(x_2,x_3|x_1)) = E_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1))+\mathbb{E}_{p(x_1,x_2)}\mathsf{KL}(p(x_3|x_1,x_2) \parallel q(x_3|x_1,x_2))$$

We can then write:

$$\begin{aligned}\mathsf{KL}(p(x_1,x_2,x_3) \parallel q(x_1,x_2,x_3)) =& \mathsf{KL}(p(x_1) \parallel q(x_1)) + \mathbb{E}_{p(x_1)}\mathsf{KL}(p(x_2|x_1) \parallel q(x_2|x_1)) \\ &+ \mathbb{E}_{p(x_1,x_2)}\mathsf{KL}(p(x_3|x_1,x_2) \parallel q(x_3|x_1,x_2))\end{aligned}$$

We generalize to $p(x_1,x_2,...x_m)$.

**Lemma 4 (KL chain rule)**

$$\begin{aligned}\mathsf{KL}(p(x_1,x_2,...x_m) \parallel q(x_1,x_2,...x_m)) =& \mathsf{KL}(p(x_1) \parallel q(x_1)) \\ &+ \sum_{j=1...m-1} \mathbb{E}_{p(x_1,x_2...x_j)}\mathsf{KL}(p(x_{j+1}|x_1,x_2...x_j) \parallel q(x_{j+1}|x_1,x_2...x_j))\end{aligned}$$

**Proof.** We use the chain rule for $p(x_1,x_2...x_m) = p(x_1).p(x_2|x_1)......p(x_{j+1}|x_1,x_2...x_j)...p(x_m|x_1,x_2...x_{m-1})$ and similarly for $q(x_1,x_2...x_m)$.

$$\log\left(\frac{p(x_1).p(x_2|x_1)......p(x_{j+1}|x_1,...x_j)...p(x_m|x_1,...x_{m-1})}{q(x_1).q(x_2|x_1)......q(x_{j+1}|x_1,...x_j)...q(x_m|x_1,...x_{m-1})}\right) = \log\left(\frac{p(x_1)}{q(x_1)}\right) + \sum_{j=1...m-1}\log\left(\frac{p(x_{j+1}|x_1,...x_j)}{q(x_{j+1}|x_1,...x_j)}\right)$$

$$\mathsf{KL}(p(x_1,x_2,...x_m) \parallel q(x_1,x_2,...x_m)) = \sum_{x_1,x_2...x_m} p(x_1,x_2...x_m).\log\left(\frac{p(x_1,x_2...x_m)}{q(x_1,x_2...x_m)}\right)$$

For the first term of the sum:

$$\sum_{x_1,x_2...x_m} p(x_1,x_2...x_m).\log\left(\frac{p(x_1)}{q(x_1)}\right) = \sum_{x_1}\log\left(\frac{p(x_1)}{q(x_1)}\right)\sum_{x_2...x_m} p(x_1,x_2...x_m)$$

$$= \sum_{x_1} p(x_1).\log\left(\frac{p(x_1)}{q(x_1)}\right) = \mathsf{KL}(p(x_1) \parallel q(x_1))$$

For the $j$-term of the sum:

$$\sum_{x_1,x_2...x_m} p(x_1,x_2...x_m)\log\left(\frac{p(x_{j+1}|x_1,...x_j)}{q(x_{j+1}|x_1,...x_j)}\right)$$

$$\sum_{x_1,x_2...x_j} p(x_1,x_2...x_j)\sum_{x_{j+1}...x_m} p(x_{j+1}...x_m|x_1,...x_j).\log\left(\frac{p(x_{j+1}|x_1,...x_j)}{q(x_{j+1}|x_1,...x_j)}\right)$$

$$= \sum_{x_1,x_2...x_j} p(x_1,x_2...x_j).\sum_{x_{j+1}} p(x_{j+1}|x_1,...x_j).\log\left(\frac{p(x_{j+1}|x_1,...x_j)}{q(x_{j+1}|x_1,...x_j)}\right)$$

$$= \sum_{x_1,x_2...x_j} p(x_1,x_2...x_j).\mathsf{KL}(p(x_{j+1}|x_1,...x_j) \parallel q(x_{j+1}|x_1,...x_j))$$

$$= \mathbb{E}_{p(x_1,x_2\ldots x_j)}.\mathsf{KL}(p(x_{j+1}|x_1,\ldots x_j) \parallel q(x_{j+1}|x_1,\ldots x_j))$$

Hence we can write:

$$\mathsf{KL}(p(x_1,x_2,\ldots x_m) \parallel q(x_1,x_2,\ldots x_m)) = \mathsf{KL}(p(x_1) \parallel q(x_1)) + \sum_{j=1\ldots m-1} \mathbb{E}_{p(x_1,x_2\ldots x_j)}\mathsf{KL}(p(x_{j+1}|x_1,x_2\ldots x_j) \parallel q(x_{j+1}|x_1,x_2\ldots x_j))$$

□

Pinsker's inequality is the natural link between $\mathsf{KL}(p(x_1,x_2,\ldots x_m) \parallel q(x_1,x_2,\ldots x_m))$ and the $L_1$ or variation distance $\mathsf{dist}_{L_1}(p(x_1,x_2,\ldots x_m), q(x_1,x_2,\ldots x_m))$.

**Lemma 5 (Pinsker's Inequality)** *For any probability distributions $p$ and $q$:*

$$\mathsf{dist}_{L_1}(p,q)^2 \leq 2 \cdot \mathsf{KL}(p \parallel q)$$

*where* $\mathsf{dist}_{L_1}(p,q) = \sum_x |p(x) - q(x)|$ *is the $L_1$ distance between the distributions.*

**Lemma 6** *For the probability distributions $p(x_1,x_2,\ldots x_m)$ and $q(x_1,x_2,\ldots x_m)$:*
*if* $\mathsf{KL}(p(x_1,x_2,\ldots x_m) \parallel q(x_1,x_2,\ldots x_m)) \leq \varepsilon$ *then for $j = 1\ldots m-1$:*

$$\mathsf{dist}_{L_1}(p(x_1), q(x_1)) \leq \sqrt{2\varepsilon}$$

$$\mathbb{E}_{p(x_1,x_2\ldots x_j)}\mathsf{dist}_{L_1}(p(x_{j+1}|x_1,x_2\ldots x_j), q(x_{j+1}|x_1,x_2\ldots x_j))^2 \leq 2\varepsilon$$

**Proof.** If $\mathsf{KL}(p(x_1,x_2,\ldots x_m) \parallel q(x_1,x_2,\ldots x_m)) \leq \varepsilon$, by Lemma 4 as each term is positive:
$\mathsf{KL}(p(x_1) \parallel q(x_1)) \leq \varepsilon$ and for $j = 1\ldots m-1$:

$$\mathbb{E}_{p(x_1,x_2\ldots x_j)}\mathsf{KL}(p(x_{j+1}|x_1,x_2\ldots x_j) \parallel q(x_{j+1}|x_1,x_2\ldots x_j)) \leq \varepsilon$$

For the first term $\mathsf{KL}(p(x_1) \parallel q(x_1)) \leq \varepsilon$, by Pinsker's inequality Lemma 5, $\mathsf{dist}_{L_1}(p(x_1), q(x_1)) \leq \sqrt{2\varepsilon}$. For the $j$-th term:

$$\mathbb{E}_{p(x_1,x_2\ldots x_j)}\mathsf{KL}(p(x_{j+1}|x_1,x_2\ldots x_j) \parallel q(x_{j+1}|x_1,x_2\ldots x_j)) \leq \varepsilon$$

By Pinsker's inequality:

$$\mathbb{E}_{p(x_1,x_2\ldots x_j)}\mathsf{dist}_{L_1}(p(x_{j+1}|x_1,x_2\ldots x_j), q(x_{j+1}|x_1,x_2\ldots x_j))^2 \leq 2\varepsilon$$

□

### 4.2.4 Bounding the Probability of Illegal Moves

**Theorem 1 (Bound on Illegal Move Probability)** *If* $\mathsf{KL}(p \parallel p_\Theta) \leq \varepsilon$, *then the expectation over a prefix $h$ taken from the distribution $p$ of the probability that the transformer generates a legal next move,* $\mathbb{E}_{p(x_1,x_2\ldots x_j)}Prob_{p_\Theta}[x_{j+1} \in \mathcal{L}]$ *is at least* $1 - \sqrt{2\varepsilon}$.

**Proof.** We use the previous lemmas with $q = p_\Theta$, the distribution learnt by the Transformer. When the prefix is empty (start of the game), by Lemma 3.:

$$\mathsf{KL}(p(x_1) \parallel p_\Theta(x_1)) \leq \varepsilon$$

Applying Pinsker's inequality (Lemma 5):

$$\mathsf{dist}_{L_1}(p(x_1), p_\Theta(x_1)) \leq \sqrt{2\varepsilon}$$

Let $\mathcal{L}$ be the set of all legal moves, i.e. $T(s_0, x_1) = 1$ where $s_0$ is the initial state, and $\mathcal{I}$ be the set of illegal moves. Since the distribution $p$ assigns zero probability to illegal moves:

$$p(x_1 \in \mathcal{I}) = 0$$
$$p(x_1 \in \mathcal{L}) = 1$$

The $L_1$ distance can be split as:

$$\begin{aligned}\mathsf{dist}_{L_1}(p(x_1), p_\Theta(x_1)) &= \sum_{x_1 \in \mathcal{L}} |p(x_1) - p_\Theta(x_1)| + \sum_{x_1 \in \mathcal{I}} |p(x_1) - p_\Theta(x_1)| \\ &= \sum_{x_1 \in \mathcal{L}} |p(x_1) - p_\Theta(x_1)| + \sum_{x_1 \in \mathcal{I}} p_\Theta(x_1) \\ &\geq \sum_{x_1 \in \mathcal{I}} p_\Theta(x_1) = \mathbb{P}_{p_\Theta}[x_1 \text{ is an illegal move}]\end{aligned}$$

Therefore: $\boldsymbol{Prob}_{p_\Theta}[x_1 \text{ is an illegal move}] \leq \mathsf{dist}_{L_1}(p(x_1), p_\Theta(x_1)) \leq \sqrt{2\varepsilon}$
Thus $\boldsymbol{Prob}_{p_\Theta}[x_1 \in \mathcal{L}] \geq 1 - \delta$ where $\delta = \sqrt{2\varepsilon}$.

Consider the case where the prefix $h = x_1, \ldots x_j$. By Lemma 6, $E_{p(x_1, x_2 \ldots x_j)} \mathsf{KL}(p(x_{j+1}|x_1, x_2 \ldots x_j) \parallel q(x_{j+1}|x_1, x_2 \ldots x_j)) \leq \varepsilon$ and by Pinsker inequality 5:

$$\mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))^2/2 \leq \mathsf{KL}(p(x_{j+1}|x_1, x_2 \ldots x_j) \parallel q(x_{j+1}|x_1, x_2 \ldots x_j))$$

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \mathsf{KL}(p(x_{j+1}|x_1, x_2 \ldots x_j) \parallel q(x_{j+1}|x_1, x_2 \ldots x_j)) \leq \varepsilon$$

Hence:

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))^2 \leq 2\varepsilon$$

By Jensen's inequality

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))]^2$$

$$\leq \mathbb{E}_{p(x_1, x_2 \ldots x_j)} \mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))^2 \leq 2\varepsilon$$

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))] \leq \sqrt{2\varepsilon}$$

As in the previous case, $\boldsymbol{Prob}_{p_\Theta}[x_{j+1} \in \mathcal{I}] \leq \mathsf{dist}_{L_1}(p(x_{j+1}|x_1, x_2 \ldots x_j), q(x_{j+1}|x_1, x_2 \ldots x_j))$, hence

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \boldsymbol{Prob}_{p_\Theta}[x_{j+1} \in \mathcal{I}] \leq \sqrt{2\varepsilon}$$

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} (1 - \boldsymbol{Prob}_{p_\Theta}[x_{j+1} \in \mathcal{L}]) \leq \sqrt{2\varepsilon}$$

$$\mathbb{E}_{p(x_1, x_2 \ldots x_j)} \boldsymbol{Prob}_{p_\Theta}[x_{j+1} \in \mathcal{L}] \geq 1 - \sqrt{2\varepsilon}$$

The probability of an illegal move is small in expectation.

□

## 4.3 Compositions

Theorem 1 has direct implications for the compositional abilities of transformer models. It shows that if a transformer can learn to approximate the distribution of legal chess games with a small KL divergence $\varepsilon$, then the probability of generating an illegal move remains bounded by $\delta = \sqrt{2\varepsilon}$ regardless of the prefix choosen with probability $p$.

We now study the composition of moves: assume a prefix of $i$ moves taken from the distribution $p$, the Next move $x_{i+1}$ taken from the distribution $p_\Theta$. The new prefix of length $i+1$ is therefore taken from a new distribution $\tilde{p}_i$ such that:

$$\tilde{p}_i(x_1,...,x_i) = p(x_1,...,x_i)$$
$$\tilde{p}_i(x_{i+1} \mid x_1,...,x_i) = p_\Theta(x_{i+1} \mid x_1,...,x_i)$$

**Lemma 7** *For the probability distributions* $p(x_1,...x_{i+1})$ *and* $p_\Theta(x_1,...x_{i+1})$*:*

$$\mathsf{KL}(p \parallel \tilde{p}) \leq \mathsf{KL}(p \parallel p_\Theta)$$

**Proof.** We write: $p(x_1,...x_{i+1}) = p(x_1,...,x_i).p(x_{i+1} \mid x_1,...,x_i)$ and similarly for $p_\Theta$ and $\tilde{p}$.

We first develop $\mathsf{KL}(p(x_1,...,x_i) \parallel \tilde{p})$ and then $\mathsf{KL}(p \parallel p_\Theta)$:

$$\begin{aligned}
\mathsf{KL}(p(x_1,...x_{i+1}) \parallel \tilde{p}(x_1,...x_{i+1})) &= \sum_{x_1,x_2} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...x_{i+1})}{\tilde{p}(x_1,...x_{i+1})}\right) \\
&= \sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...,x_i).p(x_{i+1} \mid x_1,...,x_i)}{p(x_1,...,x_i).p_\Theta(x_{i+1} \mid x_1,...,x_i)}\right) \\
&= \sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_{i+1} \mid x_1,...,x_i)}{p_\Theta(x_{i+1} \mid x_1,...,x_i)}\right)
\end{aligned}$$

For $\mathsf{KL}(p \parallel p_\Theta)$:

$$\begin{aligned}
\mathsf{KL}(p(x_1,...x_{i+1}) \parallel p_\Theta(x_1,...x_{i+1})) &= \sum_{x_1,x_2} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...x_{i+1})}{p_\Theta(x_1,...x_{i+1})}\right) \\
&= \sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...,x_i).p(x_{i+1} \mid x_1,...,x_i)}{p_\Theta(x_1,...,x_i).p_\Theta(x_{i+1} \mid x_1,...,x_i)}\right) \\
&= \sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...,x_i)}{p_\Theta(x_1,...,x_i)}\right) \\
&+ \sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_{i+1} \mid x_1,...,x_i)}{p_\Theta(x_{i+1} \mid x_1,...,x_i)}\right)
\end{aligned}$$

Notice that

$$\begin{aligned}
\sum_{x_1,...x_{i+1}} p(x_1,...x_{i+1}) \log\left(\frac{p(x_1,...,x_i)}{p_\Theta(x_1,...,x_i)}\right) &= \sum_{x_1,...x_i} p(x_1,...x_i) \log\left(\frac{p(x_1,...,x_i)}{p_\Theta(x_1,...,x_i)}\right) \\
&= \mathsf{KL}(p(x_1,...,x_i) \parallel p_\Theta(x_1,...,x_i)) \geq 0 \qquad (4)
\end{aligned}$$

Therefore:

$$\mathsf{KL}(p \parallel p_\Theta) = \mathsf{KL}(p(x_1,...,x_i) \parallel p_\Theta(x_1,...,x_i)) + \mathsf{KL}(p \parallel \tilde{p})$$

We conclude that:

$$\mathsf{KL}(p \parallel \tilde{p}) \leq \mathsf{KL}(p \parallel p_\Theta)$$

□

We can now conclude that:

**Lemma 8** *If* $\mathsf{KL}(p \parallel p_\Theta) \leq \varepsilon$ *then* $\mathsf{dist}(p,\tilde{p}) \leq \sqrt{2\varepsilon}$

**Proof.** From the previous lemma **??**, we infer that $\mathsf{KL}(p \parallel \tilde{p}) \leq \varepsilon$ and from Pinsker's inequality 5, we infer that $\mathsf{dist}(p,\tilde{p}) \leq \sqrt{2\varepsilon}$. □

**Theorem 2 (Bound on the composition)** *If* $\mathsf{KL}(p \parallel p_\Theta) \leq \varepsilon$*, then the expectation over a prefix h taken from the distribution* $\tilde{p}$ *of the probability that the transformer generates a legal next move,* $\mathbb{E}_{\tilde{p}(x_1,\dots x_j)} p_\Theta(x_{i+1} \in \mathcal{L} \mid x_1,\dots,x_i)$*, is at least* $1 - 2\sqrt{2\varepsilon}$*.*

**Proof.** Let $f = p_\Theta(x_{i+1} \in \mathcal{L}] \mid x_1,\dots,x_i)$ hence $0 \leq f \leq 1$. We show that $\mathbb{E}_{\tilde{p}(x_1,\dots x_j)} f$ is close to 1. By theorem 1, $\mathbb{E}_{p(x_1,\dots x_j)}\, p_\Theta(x_{i+1} \in \mathcal{L} \mid x_1,\dots,x_i) \geq 1 - \sqrt{2\varepsilon}$. Let us show that:

$$\mid \mathbb{E}_{p(x_1,x_2\dots x_j)} f - \mathbb{E}_{\tilde{p}(x_1,x_2\dots x_j)} f \mid \; \leq \mathsf{dist}(\tilde{p},p)$$

Let $S = \{(x_1,x_2\dots x_j) \; : \; p(x_1,x_2\dots x_j) \geq \tilde{p}(x_1,x_2\dots x_j)$. Then

$$\mid \mathbb{E}_{p(x_1,\dots x_j)} f - \mathbb{E}_{\tilde{p}(x_1,\dots x_j)} f \mid = \sum_{(x_1,\dots x_j)\in S} f \cdot ((p(x_1,\dots x_j) - \tilde{p}(x_1,x_2\dots x_j)) + \sum_{(x_1,\dots x_j)\notin S} f \cdot ((\tilde{p}(x_1,\dots x_j) - p(x_1,\dots x_j))$$

$$\mid \mathbb{E}_{p(x_1,x_2\dots x_j)} f - \mathbb{E}_{\tilde{p}(x_1,x_2\dots x_j)} f \mid = f \cdot \mathsf{dist}(p,\tilde{p})$$

By lemma 8, $\mathsf{dist}(p,\tilde{p}) \leq \sqrt{2\varepsilon}$ and by theorem 1, $\mathbb{E}_{p(x_1,x_2\dots x_j)} f \geq 1 - \sqrt{2\varepsilon}$. Therefore

$$\mathbb{E}_{\tilde{p}(x_1,x_2\dots x_j)} p_\Theta(x_{i+1} \in \mathcal{L}] \mid x_1,\dots,x_i) \geq 1 - 2\sqrt{2\varepsilon}$$

□

## 5 Experimental Setup

To empirically validate our theoretical analysis, we conducted a series of experiments with transformer models trained on chess games, following methodologies from recent work [5, 14, 19, 28]. This section details our experimental methodology, including model architecture, training procedures, and evaluation metrics, which build upon established approaches in neural network analysis [27, 17].

### 5.1 Model Architecture

We trained four separate transformer models with identical architectures but different levels of noise in their training data. This approach allowed us to investigate how training data quality affects compositional capabilities. The models share the following architectural characteristics:

The chosen sequence length of 1024 tokens is more than sufficient for our task, as most chess games require significantly fewer tokens. A typical chess game rarely exceeds 80 moves (160 ply), and each move in PGN notation requires at most 7 tokens (including move number and punctuation). This ensures that our models can process complete games during both training and inference.

| Parameter | Value |
|---|---|
| Transformer layers | 8 |
| Attention heads per layer | 8 |
| Embedding dimension | 512 |
| Feed-forward dimension | 2048 |
| Max sequence length | 1024 tokens |
| Vocabulary size | 32 tokens |
| Total parameters | $\approx$ 1M |

Table 1: Model architecture specifications

## 5.2 Dataset

Our dataset was derived from the Lichess open database of chess games, with the following filtering criteria:

- Games played by players with Elo ratings above 2000
- Standard chess games (no variants like Chess960)
- Games with at least 20 moves
- Games without adjudication by timeout or abandonment

After filtering, our base dataset consisted of approximately 1 million games, which we split into training (80%), validation (10%), and test (10%) sets.

### 5.2.1 Noise Injection

To study how model performance degrades with imperfect training data, we created four versions of the dataset with different levels of noise:

- **Clean (0%)**: The original dataset with no modifications
- **Low Noise (5%)**: 5% of moves randomly replaced with alternative legal moves
- **Medium Noise (50%)**: 50% of moves randomly replaced with alternative legal moves
- **High Noise (80%)**: 80% of moves randomly replaced with alternative legal moves

Importantly, we only replaced moves with other legal moves, ensuring that all games in the dataset remained syntactically valid. This approach tests the model's ability to learn strategic patterns rather than just the rules of the game.

### 5.2.2 Random Move Selection Mechanism

The generation of random moves in our training data creation process follows a uniform approach among legal moves. For each game position where a move must be replaced, we proceed as follows:

1. From the current board position $s$, we compute the set $\mathcal{L}(s)$ of all available legal moves
2. Each move $m \in \mathcal{L}(s)$ has equal probability of being selected: $P(m) = \frac{1}{|\mathcal{L}(s)|}$
3. The new move $m'$ is drawn according to this uniform discrete distribution

| Hyperparameter | Value |
|---|---|
| Optimizer | AdamW |
| Batch size | 4 |
| Initial learning rate | 1e-3 |
| Final learning rate | 1e-5 |
| Learning rate schedule | Linear decay |
| Warmup steps | 500 |
| Number of epochs | 5 |
| Weight decay | 0.01 |

Table 2: Training hyperparameters

4. The game continues with this new move, and the subsequent position is computed accordingly

This uniform sampling method guarantees that:

- All legal moves have the same probability of being chosen, eliminating any strategic bias
- The syntactic and semantic validity of games is preserved
- The level of "noise" injected can be precisely controlled by the percentage of moves replaced

Mathematically, if $G = (m_1, m_2, \ldots, m_k)$ represents an original game and $G' = (m'_1, m'_2, \ldots, m'_k)$ the modified game, then for each replaced move $m'_i$:

$$m'_i \sim \text{Uniform}(\mathcal{L}(s_{i-1}))$$

where $s_{i-1}$ is the board position after moves $m'_1, \ldots, m'_{i-1}$.

### 5.3 Training Procedure

All models were trained using a standard language modeling objective, minimizing the cross-entropy loss for next token prediction. We used the following hyperparameters:

Models were trained on 1 NVIDIA L4 GPU, with training time varying from 18 to 24 hours depending on the dataset version.

### 5.4 Evaluation Metrics

We evaluated our models using three complementary approaches to assess their compositional capabilities:

#### 5.4.1 Legal Move Rate

The most basic measure of compositional understanding is the ability to generate legal chess moves. We evaluated this by:

- Providing the model with a game prefix from the test set
- Generating the next move using temperature sampling ($T = 0.8$)
- Checking if the generated move is legal in the current board position

We report the percentage of legal moves generated across 10,000 different positions from the test set.

| **Position Type** | **Clean (0%)** | **Low Noise (5%)** | **Medium Noise (50%)** | **High Noise (80%)** |
|---|---|---|---|---|
| Opening (moves 1-10) | 98.4% | 99.7% | 99.9% | 100.0% |
| Middlegame (moves 11-30) | 98.3% | 98.9% | 99.3% | 99.6% |
| Endgame (moves 31+) | 94.7% | 95.5% | 96.6% | 97.9% |
| All positions | 97.0% | 98.0% | 98.6% | 99.2% |

Table 3: Legal move rates for models trained with different noise levels

#### 5.4.2 Tactical Problem-Solving

To assess deeper compositional understanding, we tested the models on tactical puzzles from Lichess's puzzle database. For a puzzle of Elo rating $e$, we define the puzzle-solving skill at rate $\alpha$ as:

$$\mathbb{P}_x[\text{Next}(x) = y \mid (x, y) \in \text{puzzles\_PGN}_e] \geq \alpha \tag{5}$$

where Next is the model's next-move prediction function. We evaluated performance on puzzles across different difficulty levels, from 1200 to 2400 Elo.

#### 5.4.3 Representation Analysis

Following methods established in the representational analysis literature [11, 17, 15, 27], we employed probing techniques to analyze the internal representations of our models:

- We extracted activations from different layers of the transformer
- We trained linear classifiers on these activations to predict various chess-relevant features (piece positions, material balance, king safety, etc.)
- We assessed how well different model layers encode these chess concepts

## 6 Results and Analysis

We present the results of our experiments, focusing on how well transformer models learn to compose chess moves and how this ability relates to our theoretical analysis.

### 6.1 Legal Move Generation

Table 3 shows the legal move rates achieved by models trained on datasets with different noise levels. The legal move rate measures the percentage of generated moves that comply with chess rules, effectively testing the models' ability to compose valid game sequences.

These results demonstrate several key findings:

- Even with no explicit knowledge of chess rules, transformers can learn to generate legal moves with high accuracy
- Performance degrades gracefully as training data quality decreases
- Legal move rates are consistently higher in opening positions compared to middlegame and endgame positions

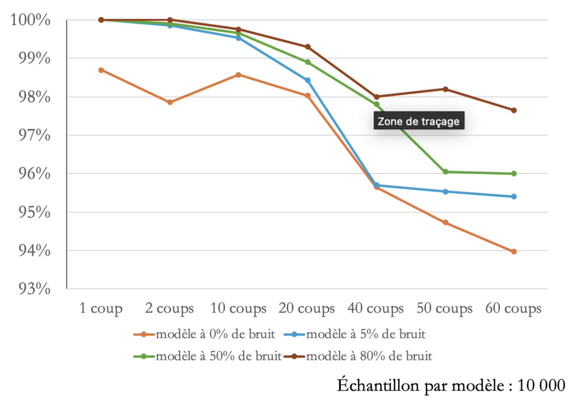


Figure 2: Illegal move rate as a function of move number for games generated by models trained with different noise levels. Error bars represent 95% confidence intervals across 1000 generated games.

For the model trained on clean data, we observe the remarkable result:

$$\mathbb{P}[M(s,s') \implies R(s,s')] \approx 0.987 \tag{6}$$

where $M(s,s')$ represents a move predicted by the transformer from position $s$ to position $s'$, and $R(s,s')$ is the relation of legal chess moves. This high probability confirms our theoretical prediction that transformers can effectively learn to compose legal move sequences, consistent with findings in related domains [19, 5, 28, 14].

## 6.2 Effect of Sequential Depth on Error Rates

A critical question is whether errors accumulate as the game progresses. Figure 2 shows the illegal move rate as a function of move number for games generated by our models.

Remarkably, the error rate increases only slightly with game length, even for models trained on noisy data. This supports our theoretical result that the probability of illegal moves remains bounded regardless of sequence length, provided the model has learned the distribution of legal chess games with sufficient accuracy.

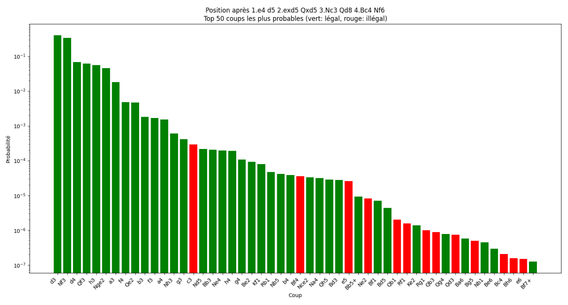
## 6.3 Move Distribution Analysis

To understand how transformers represent chess knowledge, we visualized the distribution of moves generated by our models at different game stages. Figure 3 shows heatmaps of move probabilities for a specific historical game (Scachs d'amor, one of the earliest recorded chess games from the 15th century).

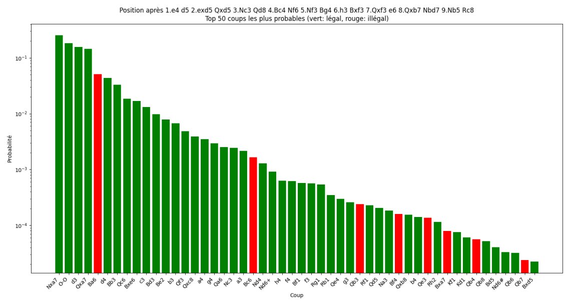
The heatmaps reveal several interesting patterns:

- In early game positions, probability mass is distributed across many plausible moves
- As the game progresses, the distribution becomes more concentrated, focusing on tactically relevant moves
- The model assigns negligible probability to illegal moves across all game stages

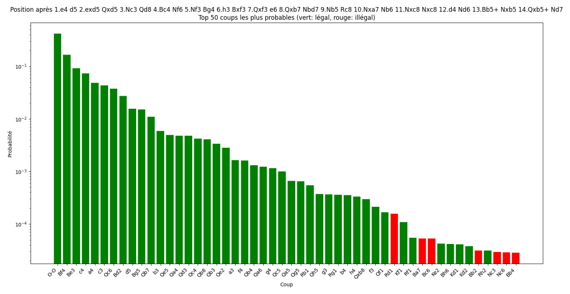
To assess how the model generalizes to novel positions, we also examined move distributions on randomly selected positions from the test set. Figure 4 shows these distributions at different game stages.



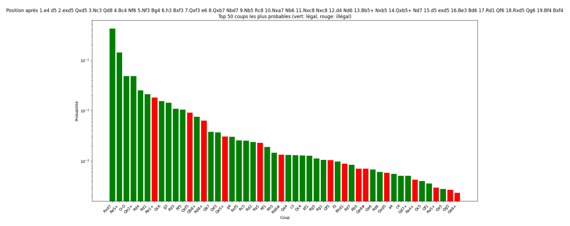
(a) Distribution at 5th move



(b) Distribution at 10th move



(c) Distribution at 15th move



(d) Distribution at 20th move

Figure 3: Distribution of move probabilities at different stages of the Scachs d'amor game. Green bars represent legal moves while red bars represent illegal moves. Note how the distribution becomes increasingly concentrated as the game progresses, reflecting the more constrained set of reasonable moves in complex positions.

### 6.4 In-Distribution vs. Out-of-Distribution Performance

We also compared the model's performance on in-distribution (ID) positions (similar to those in the training set) versus out-of-distribution (OOD) positions (significantly different from the training set). Figure 6 shows the average move distributions for these two categories at different game stages.

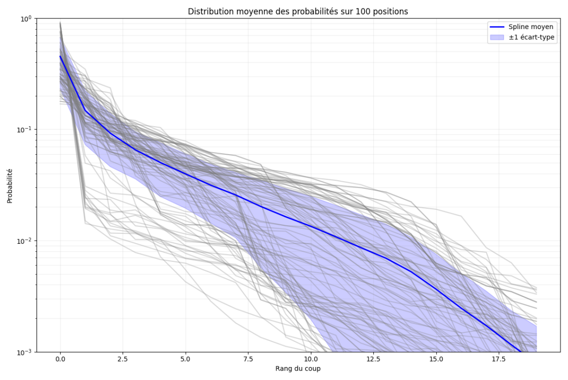
The key observations from this analysis are:

- Legal move rates remain high for both ID and OOD positions, though they are somewhat lower for OOD positions
- The gap between ID and OOD performance widens as the game progresses
- Even in OOD positions, the model focuses probability mass on strategically reasonable moves

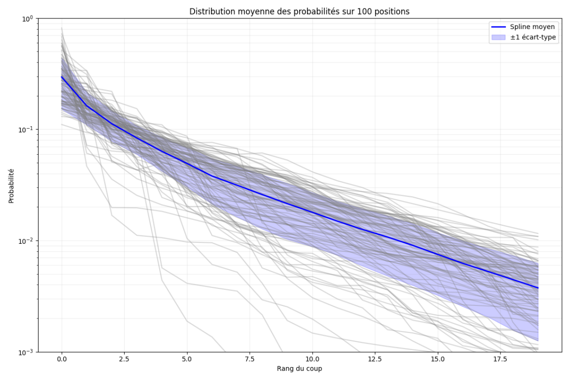
These findings support our theoretical analysis: the model has learned to approximate the distribution of legal chess sequences with sufficient accuracy that it maintains compositional validity even in unfamiliar positions, demonstrating similar characteristics to those observed in other pattern learning studies [26, 29, 23].

## 7 Conclusion

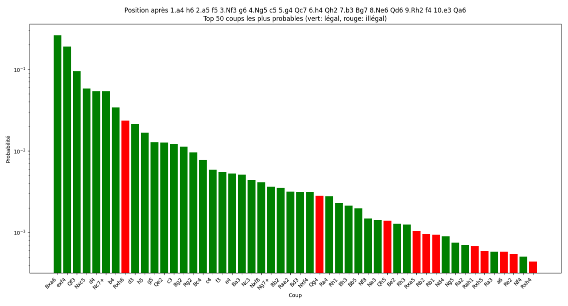
In this paper, we have addressed a fundamental question about transformer models: how well can they compose functions or relations despite theoretical limitations? Our investigation, focused on chess as a benchmark domain, reveals several important insights:



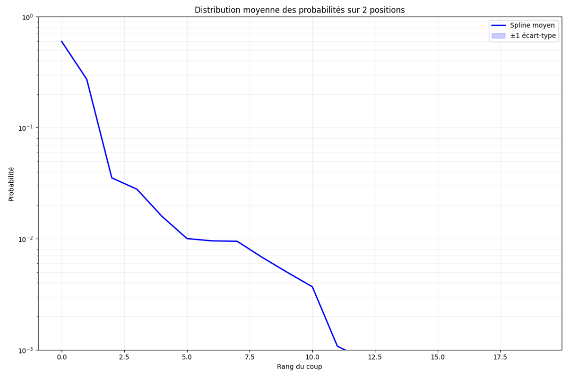
(a) Distribution at second move



(b) Distribution at sixth move



(c) Distribution at tenth move



(d) Distribution at endgame position

Figure 4: Move distributions for randomly selected positions from the test set. These visualizations demonstrate the model's ability to generate appropriate probability distributions even for positions it hasn't seen during training.

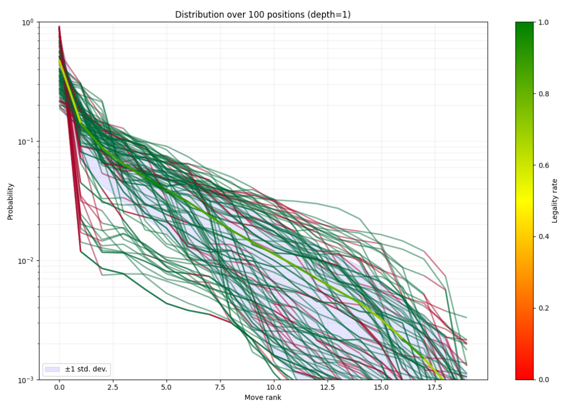
1. **Theoretical analysis**: We demonstrated that while worst-case analysis suggests limitations in functional composition for transformers, the distributional properties of real-world data significantly improve compositional capabilities. We provided bounds on the probability of illegal moves that remain constant regardless of sequence length.

2. **Empirical validation**: Our experiments confirmed the theoretical predictions, showing that transformers can learn to generate legal chess moves with high accuracy (98.7% for our best model) despite never being explicitly taught the rules of chess.

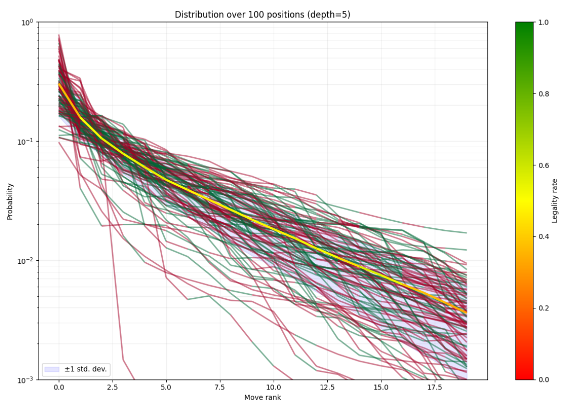
3. **Error accumulation**: Both theory and experiments indicate that composition errors do not accumulate significantly during extended sequences, explaining why transformers can maintain coherence over long outputs.

4. **Distribution learning**: The models effectively learned the distribution of legal chess games, with error rates closely matching our theoretical predictions based on the KL divergence between learned and true distributions.

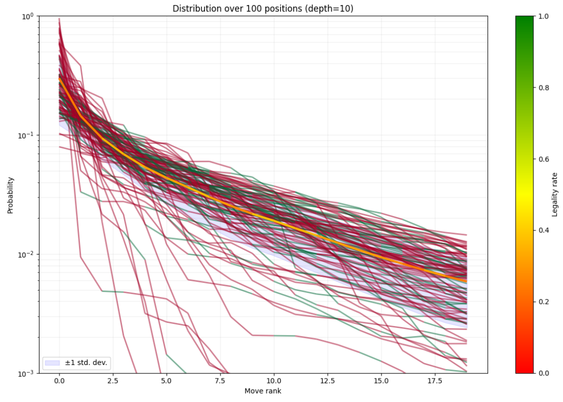
Our work bridges the gap between theoretical worst-case analysis and empirical observations of composition in transformer models [21, 9]. It suggests that the success of these models in sequential tasks stems from their ability to learn accurate distributions over compositional structures, rather than from



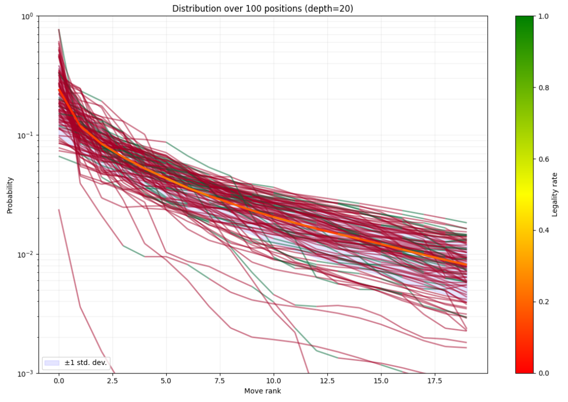
(a) Average distribution at 5th move



(b) Average distribution at 10th move



(c) Average distribution at 15th move



(d) Average distribution at 20th move

Figure 5: Comparison of average move probability distributions at different game stages.

explicitly representing compositional rules, aligning with recent findings in emergent behavior studies [19, 2].

These findings have implications beyond chess, offering insights into how transformers handle composition in other domains such as natural language [30, 3], mathematics [26], and programming [22]. The theoretical framework we developed can be applied to analyze compositional capabilities in these domains as well, extending work on latent representation analysis [11, 12].

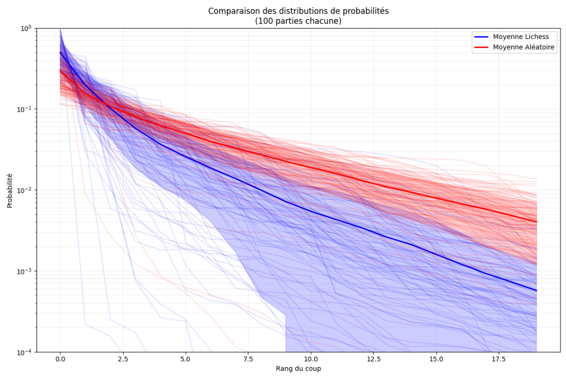
Future work could explore how these results extend to other types of compositional tasks, investigate the internal mechanisms that enable successful composition [28, 14], and develop methods to enhance compositional generalization in transformer models [29, 23].

## References

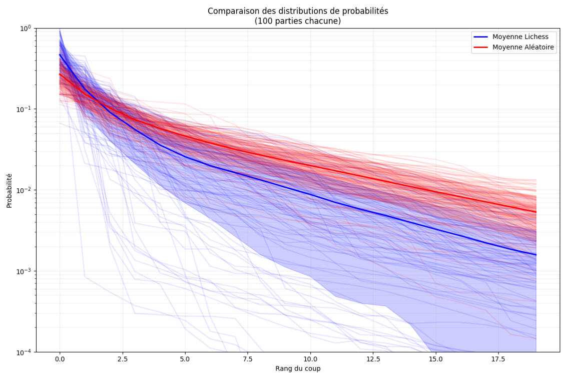
[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] Sanjeev Arora and Archit Goyal. Skill-mix: A flexible and expandable family of evaluations for ai models. *arXiv preprint arXiv:2310.17567*, 2023.

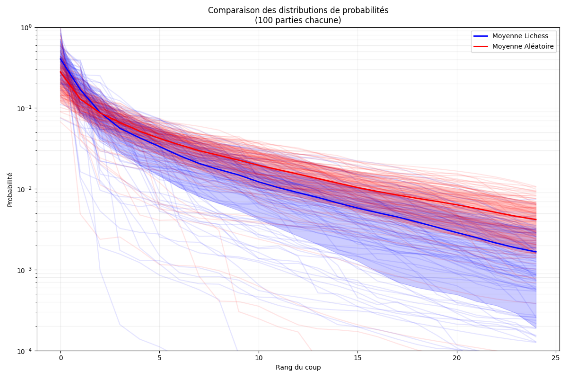
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.



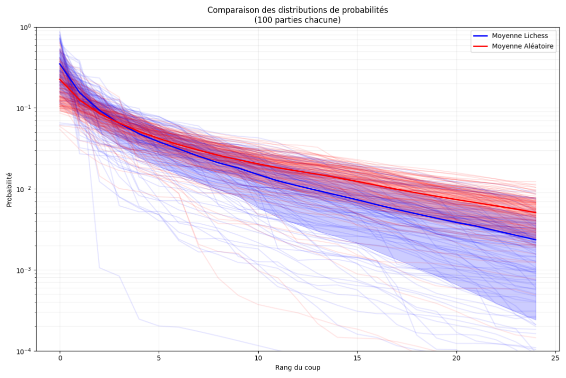
(a) Average distribution at 5th move



(b) Average distribution at 10th move



(c) Average distribution at 15th move



(d) Average distribution at 20th move

Figure 6: Comparison of average move probability distributions for in-distribution (solid lines) and out-of-distribution positions (dashed lines) at different game stages. While the model performs better on ID positions, it maintains surprisingly good performance on OOD positions, particularly in terms of legal move generation.

[4] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48:1–12, 2021.

[5] Nicholas Carlini. Playing chess with large language models. `https://nicholas.carlini.com/writing/2023/chess-llm.html`, 2023.

[6] Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer. In *Foundations of Computer Science (FOCS)*, 2025.

[7] Pablo Diego-Simòn, Stéphane D'Ascoli, Emmanuel Chemla, Yair Lakretz, and Jean-Rémi King. A polar coordinate system represents syntax in large language models, 2024.

[8] Pablo J. Diego-Simon, Emmanuel Chemla, Jean-Rémi King, and Yair Lakretz. Probing syntax in large language models: Successes and remaining challenges, 2025.

[9] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publication of the mathematical institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[10] Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[11] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019.

[12] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[13] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. On the limits of language generation: Trade-offs between hallucination and mode collapse, 2025.

[14] Aleksi Karvonen. Emergent world models and latent variable estimation in chess-playing language models. *arXiv preprint arXiv:2403.15498*, 2024.

[15] Marius Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2024.

[16] Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit, 2024.

[17] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[18] E. Kushilewitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[19] Kevin Li, Fernanda Viegas, Andrew K Hopkins, Hanspeter Pfister, David Bau, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[20] Lichess.org. Lichess: Free online chess. `https://lichess.org/`, 2024. Accessed: May 4, 2024.

[21] Ilan Newman and Mauricio Karchmer. The communication complexity of correlation. *SIAM Journal on Computing*, 26(1):76–82, 1997.

[22] OpenAI. GPT-4. `https://openai.com/gpt-4`, 2024.

[23] Adam Pearce, Asma Ghandeharioun, Nisreen Hussein, Nithum Thain, Martin Wattenberg, and Lucy Dixon. Do machine learning models memorize or generalize? *arXiv preprint arXiv:2308.07613*, 2023.

[24] Judea Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*, 2018.

[25] Binghui Peng, Srini Narayanan, and Christos Papadimitriou. On limitations of the transformer architecture. In *Conference on Language Modeling*, 2024.

[26] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. In *International Conference on Learning Representations (ICLR)*, 2022.

[27] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in neural information processing systems*, pages 6076–6085, 2017.

[28] Alex Templeton, Thomas Conerly, Jacob Marcus, Jack Lindsey, Trenton Bricken, Benjamin Chen, Adam Pearce, Chris Citro, Emmanuel Ameisen, Akilesh Jones, Hunter Cunningham, Nicholas L Turner, Callum McDougall, Matthew MacDiarmid, Christopher D Freeman, Ted R Sumers, Ellen Rees, Joshua Batson, Adam Jermyn, Sarah Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`, 2024. Transformer Circuits Thread.

[29] Vikram Varma, Rohan Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[31] Wikipedia. Elo rating system. `https://en.wikipedia.org/wiki/Elo_rating_system`, 2024. Accessed: May 4, 2024.

## A Worst-Case Composition Error

**Lemma 9 (Worst-Case Composition Error [25])** *Let $\mathcal{T}$ be a transformer with parameters $H$ (number of heads), $d$ (embedding dimension), and $p$ (precision). If the size of the internal representation is $|\Pi| = n \log n - R$ where $R = n \log n - H(d+1)p$, then the probability over random functions $f, g$ and input $x$ that $\mathcal{T}$ computes $f(g(x))$ incorrectly is at least $\frac{R}{3n \log n}$.*

**Proof.** The mutual information between the message $\Pi$ and the target value $f(i^*)$ given $i^*$ can be written as:

$$\begin{aligned} I[\Pi; f(i^*) \mid i^*] &= \sum_i \mathbb{P}[i^* = i] \cdot I[\Pi; f(i^*) \mid i^* = i] \\ &= \frac{1}{n} \sum_i I[\Pi; f(i)] \end{aligned}$$

assuming $i$ is uniformly distributed over the domain $\{1, 2, \ldots, n\}$. A fundamental result in information theory states that $I[\Pi; f(i^*) \mid i^*] \leq |\Pi|/n$, as the message $\Pi$ can provide at most $|\Pi|$ bits of information in total, which must be divided across $n$ possible values of $i^*$. Therefore:

$$\begin{aligned} H[f(i^*) \mid \Pi, i^*] &= H[f(i^*) \mid i^*] - I[\Pi; f(i^*) \mid i^*] \\ &= \log n - I[\Pi; f(i^*) \mid i^*] \\ &\geq \log n - |\Pi|/n \\ &= \log n - (n \log n - R)/n \\ &= \log n - \log n + R/n \\ &= R/n \end{aligned}$$

By Fano's inequality:

$$\begin{aligned} H(error) + \delta \cdot \log n &\geq H[f(i^*) \mid \Pi, i^*] \\ &\geq R/n \end{aligned}$$

Since $H(error) \leq 1$ for a binary random variable, and using a loose upper bound, we can say:

$$2\delta \cdot \log n \geq R/n$$
$$\delta \geq \frac{R}{2n\log n}$$

For a more precise bound, we can use the fact that $H(error) \leq -\delta \log \delta \leq 2\delta$ for small $\delta$, giving us:

$$\delta \cdot \log n + 2\delta \geq R/n$$
$$\delta(\log n + 2) \geq R/n$$
$$\delta \geq \frac{R}{n(\log n + 2)}$$
$$\delta \geq \frac{R}{3n\log n} \text{ (for large enough } n\text{)}$$

Thus, the probability of error is at least $\frac{R}{3n\log n}$. □

************************************

Old approach:

We now need to study the case when $q(x_1, x_2) = p_\Theta(x_1, x_2)$ and the marginal $p_\Theta(x_2 \mid x_1)$. We want to show that most $x_2$ are valid moves. Let $s_1$ the state of the board after the first move $x_1$.

The Transformer is a circuit of size $\alpha$ so let $H_\alpha$ be the set of distributions represented by such circuits. Then $|H_\alpha| = 2^\alpha$.

We write $x_1^i$ for the first move of the $i$-th sample, resulting in state $s_1^i$.

**Lemma 10**

$$\mathbb{E}_{p_\Theta(x_2 \mid x_1)}[T(s_1, x_2) = 1] > 1 - \delta$$

**Proof.**

Notice that $\mathbb{E}_{p(x_2 \mid x_1)}[T(s_1, x_2)] = 1$ as the distribution $p$ only includes valid moves.

For any distribution $q$, let $L(q) = \mathbb{E}_{q(x_2 \mid x_1)}[T(s_1, x_2)]$ and $\widehat{L}(q) = \sum_{i=1\ldots N} \mathbb{E}_{q(x_2 \mid x_1^i)}[T(s_1^i, x_2)]/n$

Using Hoeffding bound:

$$\textit{Prob}[|L(q) - \widehat{L}(q)| > \varepsilon] < 2\exp^{-2N.\varepsilon^2}$$

Consider then the event: $\exists q \in H_\alpha \ \ |L(q) - \widehat{L}(q)| > \varepsilon$. We apply the Union bound with $|H_\alpha| = 2^\alpha$:

$$\textit{Prob}[\exists q \in H_\alpha \ \ |L(q) - \widehat{L}(q)| > \varepsilon] < 2^\alpha.2\exp^{-2N.\varepsilon^2}$$

Apply to $p_\Theta$........

$$|\mathbb{E}_{p(x_2 \mid x_1)}[T(s_1, x_2) = 1] - \mathbb{E}_{p_\Theta(x_2 \mid x_1)}[T(s_1, x_2) = 1]| \leq \delta$$

Hence $\mathbb{E}_{p_\Theta(x_2 \mid x_1)}[T(s_1, x_2) = 1] > 1 - \delta$. □

**Lemma 11**

$$\mathbb{E}_{p_\Theta(x_{i+1} \mid x_1, \ldots x_i)}[T(s_i, x_{i+1}) = 1] > 1 - \delta$$

**Proof.**

□

Using these results, we can bound the probability of generating illegal moves:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

NO SENSE

**Lemma 12 (Composition Success Under Biased Distribution)** *Let $\mathcal{T}$ be a transformer with parameters such that the capacity of its internal representation accommodates the entropy $H(i)$ of the distribution of intermediate values. Then the probability of successful composition is at least $1 - \delta$, where $\delta$ depends on the relationship between the model capacity and $H(i)$.*

Alice sends the most frequent values to Bob, low error

**Proof.** Consider the conditional entropy $H[f(i^*) \mid \Pi, i^*]$ with two cases: error with probability $\delta$ and correct computation with probability $1 - \delta$:

$$\begin{aligned} H[f(i^*) \mid \Pi, i^*] &= \delta \cdot H[f(i^*) \mid \Pi, i^*, \text{error}] \\ &\quad + (1 - \delta) \cdot H[f(i^*) \mid \Pi, i^*, \text{no error}] \\ &= \delta \cdot H[f(i^*) \mid \Pi, i^*, \text{error}] \end{aligned} \tag{7}$$

Since $H[f(i^*) \mid \Pi, i^*, \text{no error}] = 0$ (there is no uncertainty when the computation is correct).
***** NO SENSE

□