BUDAPEST SEMESTERS IN MATHEMATICS
ADVANCED DIRECTED READING PROGRAM

Some Structures in Transformer Architectures

Vasudev Menon
Supervisor: Attila Jung

BSM-ADRP-2025-12 Autumn 2025

Webpage: https://drp.elte.hu/

Some Structures in Transformer Architectures

Vasudev Menon*

Abstract

Transformer models have become the backbone of modern large language models due to their unprecedented success and state-of-the-art performance across a wide range of tasks. However, their inner workings have remained difficult to interpret. In this project, we present a more interpretable construction of the decoder-only transformer model, focusing on a simplified, attention-only architecture. We show that the self-attention mechanism of a single attention head can be compactly written as a Kronecker product acting on the input embeddings, which highlights the separable nature of attention and provides a framework for interpreting information flow through zero- and one-layer transformers. Finally, beyond the attention mechanism itself, we provide a brief introduction to the phenomenon of superposition in high-dimensional vector spaces, highlighting the increasing complexities as these models scale.

1 Introduction

The transformer architecture [6] has emerged as a foundational neural network model for a wide range of tasks, most prominently in large language models (LLMs) such as OpenAI's GPT series [2]. As these models grow in size and complexity, understanding their internal computations becomes increasingly challenging. In particular, transformers can exhibit unexpected, harmful, or otherwise opaque behavior, motivating the need for systematic approaches to interpretability.

To address this, the field of mechanistic interpretability has emerged as a research program that seeks to uncover how individual components of a model, such as attention heads, neurons, or layers, contribute to its overall function. Often, mechanistic interpretability draws on mathematical tools and theory to guide its analysis. Insights from this line of work are crucial not only for improving model transparency, but also for ensuring AI alignment and safety [1].

In this project, we will explain the framework introduced by Elhage et al. [4] of Anthropic, which provides an introductory linear algebraic formulation of transformer circuits and decomposes their operations into interpretable linear and nonlinear components. By expressing attention computations in this compact linear algebraic form, we can better analyze how information propagates across sequence positions and embedding dimensions. We then explore the phenomenon of superposition, which helps explain why transformers are so complex in practice and why their strong performance can be difficult to fully understand. This lays the groundwork for mechanistic interpretability in larger models, where interactions across layers and the superposition of features give rise to complex, intertwined behaviors.

We begin section 2 with the necessary mathematical preliminaries before introducing the transformer architecture and establishing the simplified formulation that we will rely on throughout. We then express the transformer's operations using Kronecker products, showing how this notation captures the structure of self-attention and enables more interpretable analysis in section 3. Using this framework, we first examine the behavior of zero- and one-layer transformers in section 4 and section 5 respectively, then discuss how complexity rapidly increases in deeper models as interactions accumulate across layers in section 6. Finally, in section 7, we extend our focus beyond attention to the role of MLPs, discussing how they contribute to the phenomenon of superposition.

*Department of Mathematics, North Carolina State University. Email: vmenon2@ncsu.edu.

2 Preliminaries

Before formally describing the transformer architecture, we establish the mathematical objects it operates on and the structural conventions we will assume. Throughout, we restrict attention to decoder-only transformers for simplicity.

2.1 Neural Network Building Blocks

Before the transformer architecture is formalized, there are standard neural-network building blocks contained within it that are worth discussing first.

Definition 1 (Softmax)**.** *Let $z = (z_1, \dots, z_K) \in \mathbb{R}^K$ be a vector of real numbers. The* softmax function $\mathrm{softmax} : \mathbb{R}^K \to [0,1]^K$ *is defined componentwise by*

$$\mathrm{softmax}(z)_i \;:=\; \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \quad \text{for } i = 1, \dots, K.$$

Note that the output $\mathrm{softmax}(z)$ *is a probability vector; all entries are non-negative and sum to 1. Additionally, softmax is invariant under adding a constant to all entries, that is,* $\mathrm{softmax}(z + c) = \mathrm{softmax}(z)$ *for any $c \in \mathbb{R}^K$.*

Definition 2 (Embeddings)**.** *Let V be the vocabulary. Each token $t \in V$ is associated with an embedding vector $e_t \in \mathbb{R}^d$. A length-n input sequence $(t_1, \dots, t_n)$ is mapped to an embedding matrix $X \in \mathbb{R}^{d \times n}$, constructed by stacking the embedding vectors as columns:*

$$X = \begin{bmatrix} | & | & & | \\ e_{t_1} & e_{t_2} & \cdots & e_{t_n} \\ | & | & & | \end{bmatrix}$$

Definition 3 (Layer Normalization (LN))**.** *Given a vector $x \in \mathbb{R}^d$, layer normalization computes*

$$\mathrm{LN}(x) = \frac{x - \mu(x)}{\sigma(x)} \cdot \gamma + \beta,$$

where $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation across coordinates, $\gamma, \beta \in \mathbb{R}^d$ are learned parameters, and the multiplication with γ is element-wise. Layer normalization is applied independently to each sequence position.

Definition 4 (Rectified Linear Unit (ReLU))**.** *The* Rectified Linear Unit *(ReLU) is an element-wise activation function $\sigma : \mathbb{R} \to \mathbb{R}$ defined by*

$$\sigma(x) \;=\; \max(0, x).$$

For a vector $z \in \mathbb{R}^d$, ReLU is applied componentwise:

$$\sigma(z)_i \;=\; \max(0, z_i) \quad \text{for } i = 1, \dots, d.$$

Definition 5 (Multilayer Perceptron (MLP))**.** *The MLP is a position-wise feed-forward network applied independently to each sequence position:*

$$\mathrm{MLP}(x) = W_2\, \sigma(W_1 x + b_1) + b_2,$$

where σ is a nonlinear map (e.g., ReLU or GELU[1]).

In typical transformer architectures, this consists of two linear layers with a nonlinear map in between (like the above). However, in principle, one could stack additional linear layers and nonlinearities to form a deeper position-wise feed-forward network. Note that the MLP mixes information across embedding dimensions but not across sequence positions.

[1] $\mathrm{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the CDF of the standard normal distribution.

Definition 6 (Residual Connections)**.** *Given an input x and a sublayer F, transformers compute the residual update*

$$x + F(\mathrm{LN}(x)).$$

Note that here we describe the Pre-LN variant used in modern LLMs, which differs slightly from the Post-LN formulation in the original Transformer paper

Definition 7 (Self-Attention Head)**.** *Given an input matrix $X \in \mathbb{R}^{d\times n}$, a single self-attention head is specified by learned projection matrices*

$$W_Q,\, W_K,\, W_V \in \mathbb{R}^{d_k\times d}.$$

It computes

$$Q = W_Q X, \qquad K = W_K X, \qquad V = W_V X,$$

The head output $H \in \mathbb{R}^{d_k\times n}$ is computed as

$$A = \mathrm{softmax}\left(\frac{Q^\top K}{\sqrt{d_k}}\right), \qquad H = VA^\top.$$

The softmax matrix A introduces the only nonlinear map; all other operations are linear in X.

Definition 8 (Multi-Head Attention)**.** *Let $H_1, \ldots, H_h$ be the outputs of h independent self-attention heads. We concatenate these outputs along the feature dimension to form $\hat{H} \in \mathbb{R}^{(h\cdot d_k)\times n}$:*

$$\hat{H} = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_h \end{bmatrix}.$$

The Multi-Head Attention (MHA) module projects this concatenation back to the model dimension:

$$\mathrm{MHA}(X) = W_O \hat{H}$$

where $W_O \in \mathbb{R}^{d\times(h\cdot d_k)}$ is the output projection matrix.

Remark 9 (Terminology and Dimensions)**.** *The intermediate matrices Q, K, and V are often referred to as the query, key, and value matrices, respectively, while H denotes the head output. Regarding dimensions, n represents the sequence length (the number of columns in X) and d represents the model (or residual) dimension. The dimension d_k is the head dimension, which defines the rank of the projection matrices. In standard multi-head attention configurations, $d_k < d$.*

2.2 Transformer

We now formalize the full decoder-only transformer model used in LLMs. A transformer with L layers, hidden dimension d, and m attention heads per layer processes an input $X_0 \in \mathbb{R}^{d\times n}$ (where columns correspond to tokens) with residual connections using the recurrence:

$$X'_\ell = X_\ell + \mathrm{MultiHeadAttn}(\mathrm{LN}(X_\ell)),$$

$$X_{\ell+1} = X'_\ell + \mathrm{MLP}(\mathrm{LN}(X'_\ell)).$$

Each layer alternates between attention and MLP blocks, each preceded by layer normalization and wrapped in residual connections. After L layers, we define the transformer output:

$$T(X_0) \coloneqq X_L.$$

In practice, this output is passed into an unembedding matrix $W_{\text{out}} \in \mathbb{R}^{V\times d}$, where V is the vocabulary size, which are converted to probabilities via softmax:

$$P = \text{softmax}(W_{\text{out}} T(X_0)),$$

where the softmax is applied per column. In the context of text generation, the i-th column of P represents the probability distribution for the token at position $i+1$. The text generation procedure is described below in Algorithm 10.

Algorithm 10 (Text Generation in a Decoder-Only Transformer)**.**
Input: *Initial token sequence* $(t_1, \dots, t_n)$
Parameters: *Embedding matrix* $E \in \mathbb{R}^{d\times V}$, *transformer* $T(\cdot)$, *unembedding matrix* W_{out}
Output: *Generated sequence* $(t_1, \dots, t_{n+k})$

1. *Convert the input tokens to embeddings. We construct* X_0 *by selecting the columns of* E *corresponding to indices* t_i*:*
$$X_0 = \begin{bmatrix} E_{:,t_1} & E_{:,t_2} & \dots & E_{:,t_n} \end{bmatrix} \in \mathbb{R}^{d\times n}$$
2. *For step* $j = 1$ *to* k*:*
   (a) *Compute hidden states:* $X_L = T(X_0)$.
   (b) *Extract the logits for the final position (the last column of* X_L*):*
   $$z = W_{out}(X_L)_{:,end} \in \mathbb{R}^V$$
   (c) *Convert logits into a probability distribution:* $p = \text{softmax}(z)$.
   (d) *Sample the next token:* $t_{n+j} \sim p$.
   (e) *Append* t_{n+j} *to the sequence and update* X_0 *by concatenating the new embedding vector* $E_{:,t_{n+j}}$.

Remark 11 (Causal Masking)**.** *In sequence modeling applications, the map* $T : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ *is typically required to be* causal*. That is, the output vector at index* i *(the* i*-th column of the output matrix) must depend only on input columns* j *where* $j \le i$*. In the context of the attention mechanism defined above, this constraint is satisfied by forcing the attention matrix* A *to be lower triangular (i.e.,* $A_{ij} = 0$ *for* $j > i$*). While this triangular structure is essential for the valid probabilistic generation of sequences in practice, we omit it here for simplicity.*

For a more complete description of the standard transformer, we refer the reader to Vaswani et al. [6] and Brown et al. [2].

2.3 Simplified Transformer

To isolate the structure of attention and reduce architectural complexity, we introduce a *simplified transformer model*, in which feedforward networks, normalization, and scaling factors are omitted, leaving only the linear self-attention operation. This model captures the essential behavior of attention while removing nonlinearities that complicate analysis.

In particular, we make the following assumptions:

1. **No MLPs:** We omit MLPs to isolate attention dynamics.
2. **Linear maps only:** All affine transformations are linear; biases can be absorbed by augmenting input vectors with a constant component[2].

[2]Consider an affine transformation $x \mapsto Ax + b$ with weight A and bias b. We "fold" the bias b into A by defining $A' = \left(\begin{array}{c|c} A & b \\ \hline 0\dots0 & 1 \end{array}\right)$ and letting $x' = \begin{bmatrix} x \\ 1 \end{bmatrix}$. We then have $A'x' = \begin{bmatrix} Ax+b \\ 1 \end{bmatrix}$. Thus, we can recover the same affine transformation without explicitly needing the vector b.

3. **No layer normalization:** Normalization can be approximately absorbed into adjacent linear maps.
4. **No scaling:** We drop the $\sqrt{d_k}$ factor.

Definition 12 (Simplified Transformer)**.** *Under these assumptions, a single-head simplified transformer layer is*

$$\mathrm{Attn}(X) = X + W_O A(X), \quad A(X) = (W_V X)\,\mathrm{softmax}\big((W_Q X)^\top (W_K X)\big)^\top.$$

An L-layer simplified transformer is the composition

$$T^{(L)}(X) = T_L \circ T_{L-1} \circ \cdots \circ T_1(X).$$

This minimal model captures the core structure of self-attention and is sufficient for the Kronecker product formulation discussed in this project.

3 Kronecker Product

We leverage the Kronecker product [5] operation to express attention matrix computations in a concise and elegant manner.

Definition 13 (Kronecker Product)**.** *Let $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$. The* Kronecker product *of A and B, denoted $A \otimes B$, is defined as*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}.$$

In words, $A \otimes B$ is the block matrix obtained by replacing each entry a_{ij} of A with the matrix $a_{ij}B$.

Remark 14 (Kronecker Products as Linear Operators on Matrices)**.** *We treat a matrix $X \in \mathbb{R}^{d\times n}$ as an element of the vector space spanned by the standard basis $\{E_{ij}\}$ of matrix units. A Kronecker product $P \otimes Q$ is interpreted as a linear operator acting on this space according to*

$$(P \otimes Q)E_{ij} = Q\,E_{ij}\,P^\top,$$

and extended linearly. Equivalently, for all matrices X of compatible size,

$$(P \otimes Q)X = QXP^\top.$$

This convention aligns with the mixed-product identity proved in Lemma 17, and will be used throughout.

Example 15. *The following is a brief visual example of the Kronecker Product. Let*

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix}.$$

Then

$$A \otimes B = \begin{bmatrix} 1B & 2B \\ 3B & 4B \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}.$$

In fact, there are some elegant properties we can use to further simplify our linear algebraic formulation of attention.

Lemma 16 (Mixed-Product Property)**.** *Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{n\times r}$, and $D \in \mathbb{R}^{q\times s}$. Then*

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \in \mathbb{R}^{mp\times rs}.$$

Proof. We interpret the Kronecker product as a block matrix and compute using block multiplication rules. By definition,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}, \qquad C \otimes D = \begin{bmatrix} c_{11}D & c_{12}D & \cdots & c_{1r}D \\ c_{21}D & c_{22}D & \cdots & c_{2r}D \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1}D & c_{n2}D & \cdots & c_{nr}D \end{bmatrix}.$$

The (i, k)-th block of $(A \otimes B)(C \otimes D)$ is

$$\begin{aligned} \big[(A \otimes B)(C \otimes D)\big]_{ik} &= \sum_{j=1}^{n} (a_{ij}B)(c_{jk}D) \\ &= \sum_{j=1}^{n} a_{ij}c_{jk}(BD) \\ &= (AC)_{ik}(BD). \end{aligned}$$

Hence, the result is

$$(A \otimes B)(C \otimes D) = \begin{bmatrix} (AC)_{11}(BD) & (AC)_{12}(BD) & \cdots & (AC)_{1r}(BD) \\ (AC)_{21}(BD) & (AC)_{22}(BD) & \cdots & (AC)_{2r}(BD) \\ \vdots & \vdots & \ddots & \vdots \\ (AC)_{m1}(BD) & (AC)_{m2}(BD) & \cdots & (AC)_{mr}(BD) \end{bmatrix},$$

which, by definition, equals $(AC) \otimes (BD)$. □

Lemma 17 (Standard Matrix Identity)**.** *For conformable matrices M, X, N,*

$$MXN = (I \otimes M)\,(N^\top \otimes I)\,X,$$

where each Kronecker product acts naturally on the row and column spaces of X as indicated. Equivalently, the combined action of left-multiplying by M and right-multiplying by N is represented by the Kronecker product $N^\top \otimes M$.

Proof. We verify that both sides act identically on the standard basis. Let E_{ij} denote the elementary matrix with a 1 in position (i, j) and zeros elsewhere. Then any X can be written as $X = \sum_{i,j} X_{ij}E_{ij}$. Hence, it suffices to check the identity for E_{ij}. Compute the left-hand side:

$$ME_{ij}N = M(e_ie_j^\top)N = (Me_i)(N^\top e_j)^\top.$$

This produces a rank-one matrix whose columns are scalar multiples of Me_i, and whose column scalars are given by entries of $N^\top e_j$. Now examine the right-hand side. By the defining property of the Kronecker product,

$$(N^\top \otimes M)\,E_{ij}$$

acts by the same rule: it multiplies each column of E_{ij} by the corresponding scalar from $N^\top e_j$, and replaces the standard basis vector e_i with its image Me_i. Thus, the two expressions coincide entrywise for every i, j. Since the equality holds on all basis elements E_{ij} and both sides are linear in X, it holds for all X. □

Claim 18. *Let $X \in \mathbb{R}^{d \times n}$, and let*

$$W_V \in \mathbb{R}^{d_k \times d}, \qquad W_O \in \mathbb{R}^{d \times d_k}, \qquad A \in \mathbb{R}^{n \times n}$$

be the usual value, output and attention matrices for a single head. Define

$$V := W_V X \in \mathbb{R}^{d_k \times n}, \qquad R := V A^\top \in \mathbb{R}^{d_k \times n}, \qquad H := W_O R \in \mathbb{R}^{d \times n}.$$

Then the head output H can be written as a single Kronecker action on X:

$$H \;=\; (A \otimes (W_O W_V))\, X$$

where the operator $A \otimes (W_O W_V)$ acts on matrices by the rule $(P \otimes Q)\, X = Q\, X\, P^\top$. Equivalently (reading the ordinary matrix product),

$$H \;=\; W_O W_V X A^\top.$$

Proof. First note the three elementary operator identities (all verified column-wise):

- $(I_n \otimes W_V)X = W_V X = V$ (apply W_V to each column of X),
- $(A \otimes I_{d_k})V = V A^\top = R$ (mix columns of V according to A),
- $(I_n \otimes W_O)R = W_O R = H$ (apply W_O to each column of R).

Chaining these three operators gives the representation

$$H = (I_n \otimes W_O)\,(A \otimes I_{d_k})\,(I_n \otimes W_V)\,X.$$

Now apply Lemma 16:

$$(P_1 \otimes Q_1)(P_2 \otimes Q_2) = (P_1 P_2) \otimes (Q_1 Q_2).$$

Using this twice,

$$(I_n \otimes W_O)(A \otimes I_{d_k})(I_n \otimes W_V) = (I_n A I_n) \otimes (W_O I_{d_k} W_V) = A \otimes (W_O W_V).$$

Therefore,

$$H = (A \otimes (W_O W_V))\, X,$$

which is exactly the identity. Writing this back in ordinary matrix multiplication yields the equivalent form $H = W_O W_V\, X\, A^\top$, as claimed. □

Finally, recall from Definition 12 that

$$\mathrm{Attn}(X) = X + W_O\,(W_V X)\, \mathrm{softmax}\big((W_Q X)^\top (W_K X)\big)^\top .$$

Denoting

$$A := \mathrm{softmax}\big((W_Q X)^\top (W_K X)\big)\,,$$

our previous derivation shows that this transformation can be expressed as a single Kronecker operator acting on X:

$$H = (A \otimes (W_O W_V))\, X.$$

Thus, the complete attention formula can be equivalently written as

$$\mathrm{Attn}(X) = X + (A \otimes W_O W_V)X.$$

where A governs interactions across the sequence dimension and $W_O W_V$ acts across the feature dimension. The Kronecker product $(A \otimes W_O W_V)$ therefore captures the separable yet coupled nature of attention, a key insight for interpretability.

3.1 Virtual Weights and Tensor Representations

While the Kronecker product formulation describes how attention acts on the input matrix X, it is often useful to analyze the *effective* linear transformation imposed by a circuit, independent of the specific input sequence. We call these *virtual weights*.

To represent these weights, we lift the smaller weight matrices (which operate on the embedding dimension d) into the full sequence space (operating on $n \times d$) using the Kronecker product with the identity matrix.

Definition 19 (Virtual Weight Construction)**.** *Let $W \in \mathbb{R}^{d\times d}$ be a weight matrix operating on the embedding space. The virtual weight $\tilde{W}$ operating on the full sequence space is given by:*

$$\tilde{W} = I_n \otimes W = \begin{bmatrix} W & 0 & \dots & 0 \\ 0 & W & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W \end{bmatrix} \in \mathbb{R}^{nd\times nd}$$

where I_n is the identity matrix of size $n \times n$. This represents applying the same transformation W locally at every token position.

Remark 20 (Effective OV Weight)**.** *Using this notation, we can express the effective output-value weight for a specific head h as:*

$$\tilde{W}^h_{OV} = I_n \otimes (W^h_O W^h_V).$$

Crucially, the full attention operation derived in section 3, $(A \otimes W_{OV})$, can be factored into a "move" term and a "process" term:

$$A \otimes W_{OV} = \underbrace{(A \otimes I_d)}_{\textit{Move}}\underbrace{(I_n \otimes W_{OV})}_{\textit{Process}}.$$

This confirms that the OV circuit acts purely on semantic information (the Process *term) before the attention pattern moves that information between positions (the* Move *term).*

4 Zero-Layer Transformers

Before analyzing attention, we will first consider the simplest case: a *zero-layer transformer*. This model consists solely of a token embedding and an unembedding, without any attention or feedforward layers. Formally, given a token sequence represented as one-hot vectors, the zero-layer transformer predicts logits as $T = W_U W_E$. Using our Kronecker notation on the full sequence, this operation is:

$$T_{\text{zero}}(x) = (I_n \otimes W_U W_E)x$$

Remark 21 (The Bigram Model)**.** *Since there are no attention layers ($A\otimes \dots$), the model cannot incorporate information from other tokens. The operator $I_n \otimes W_U W_E$ maps each token independently to a prediction for the next token. Thus, a zero-layer transformer acts purely as a bigram model: it predicts $P(t_{i+1}|t_i)$ based on static statistics of adjacent token pairs.*

5 One-Layer Attention-Only Transformers

We now introduce a single layer of attention. We claim that one-layer attention-only transformers can be understood as an ensemble of a bigram model and several "skip-trigram" models.

The goal of this section is to rigorously show this correspondence by converting the raw weights of a transformer into interpretable tables of probability adjustments.

To understand the information flow end-to-end, we look at the full operation of the transformer, mapping one-hot tokens to output logits. We can express this as a composition of three linear operators acting on the sequence space: the Embedding, the Attention Layer, and the Unembedding.

Let x be the input represented as a concatenation of one-hot vectors. The model operation $T(x)$ is:

$$T(x) = \underbrace{(I \otimes W_U)}_{\text{Unembed}} \cdot \underbrace{\left((I \otimes I) + \sum_{h=1}^{H} (A^h \otimes W_O^h W_V^h) \right)}_{\text{Attention Layer}} \cdot \underbrace{(I \otimes W_E)}_{\text{Embed}} \cdot x$$

where $(I \otimes I)$ represents the residual connection and $A^h \in \mathbb{R}^{n \times n}$ denotes the attention pattern matrix for head h (which depends on x via the softmax of query-key dot products).

Proposition 22 (Path Expansion)**.** *We can distribute the terms to see the individual paths. Using the Mixed-Product Property of the Kronecker product ($(P \otimes Q)(R \otimes S) = PR \otimes QS$), as shown in Lemma 16, we calculate the contribution of each term.*

1. ***The Direct Path Term:***

$$(I \otimes W_U)(I \otimes I)(I \otimes W_E) = (I \cdot I \cdot I) \otimes (W_U \cdot I \cdot W_E) = (I \otimes W_U W_E)$$

2. ***The Attention Head Terms:***

$$(I \otimes W_U)(A^h \otimes W_O^h W_V^h)(I \otimes W_E) = (I \cdot A^h \cdot I) \otimes (W_U \cdot W_O^h W_V^h \cdot W_E)$$

Combining these gives us the full Path Expansion equation:

$$T(x) = \underbrace{(I \otimes W_U W_E)x}_{\textit{Bigram Path}} + \sum_{h=1}^{H} \underbrace{(A^h \otimes W_U W_O^h W_V^h W_E)x}_{\textit{Skip-Trigram Paths}}$$

This expansion reveals that the model is a sum of interpretable terms. The direct path $(I \otimes W_U W_E)$ corresponds to the Zero-Layer Bigram model derived previously. The attention terms, however, capture complex dependencies involving the attention pattern A^h.

To interpret these dependencies, we observe that the matrices involved in the attention head terms can be grouped into two distinct circuits that operate directly on the vocabulary space.

Definition 23 (The QK and OV Circuits)**.** *To understand how a head processes information, we separate its operation into two distinct mechanisms acting on the vocabulary space: determining* which *token to read from (Selection), and determining* what *information to transmit (Transmission).*

1. ***The Query-Key (QK) Circuit (The Selector):***

$$W_{QK}^h = W_E^\top W_Q^{h\top} W_K^h W_E \in \mathbb{R}^{n_{vocab} \times n_{vocab}}$$

This matrix governs the attention pattern A^h. It computes a compatibility score between every pair of tokens. Specifically, the bilinear form $x_i^\top W_{QK}^h x_j$ determines the "relevance" of source token x_j to destination token x_i.

Note that attention weights are these scores passed through a softmax:

$$A_{ij}^h = \text{softmax}_j(x_i^\top W_{QK}^h x_j).$$

We say the head ***attends*** *to token x_j if the resulting weight A_{ij}^h is large.*

2. ***The Output-Value (OV) Circuit (The Transmitter):***

$$W_{OV}^h = W_U W_O^h W_V^h W_E \in \mathbb{R}^{n_{vocab} \times n_{vocab}}$$

This matrix describes the effect of a source token on the output, conditional *on it being attended to. If the head attends solely to a source token x_j (i.e., $A_{ij}^h \approx 1$), the contribution to the output logits is exactly $W_{OV}^h x_j$. Therefore, intuitively, this matrix answers the question: "If we look at token x_j, what should we predict next?"*

Remark 24 (Interpretation of Circuits)**.** *By collapsing the internal weights, we can interpret these circuits intuitively:*

- ***The QK Circuit*** *determines* where to look*. A large positive entry* $(W_{QK})_{u,v}$ *means that when the current token is u, the head prefers to attend to previous occurrences of token v.*
- ***The OV Circuit*** *determines* what to write*. A large positive entry* $(W_{OV})_{u,v}$ *means that if the head attends to token u, it increases the probability of predicting token v next.*

This separation allows us to view the non-linear transformer as an interaction between two linear components. The QK circuit selects a "source" token based on the current "destination" token, and the OV circuit predicts an "out" token based on that source. Thus, instead of abstract weights, we obtain relationships between tokens. Together, the three tokens involved form a "skip-trigram" of the form:

$$[\text{source}] \ \dots \ [\text{destination}] \rightarrow [\text{out}]$$

Theorem 25 (Transformer as an Ensemble)**.** *A one-layer attention-only transformer acts as an ensemble of heuristics.*

- *The Bigram Model (Direct Path) predicts* $B \rightarrow C$*.*
- *The Skip-Trigrams (Head Paths) predict that if we are at* B *and attended to* A*, the next token is likely* C*.*

Proof. Recall the path expansion equation:

$$T(x) = (I \otimes W_U W_E)x + \sum_{h=1}^{H} (A^h \otimes W_{OV}^h)x$$

To understand the contribution of each term to the logits at position i (predicting token t_{i+1}), we analyze the two components separately. The first term, $(I \otimes W_U W_E)$, is block-diagonal, meaning it operates independently on each position. Consequently, at position i, the output depends solely on the input at that same position:

$$\text{Logits}_{\text{direct}} = W_U W_E \cdot x_i$$

This mirrors the definition of a Bigram Model, where the probability of the next token relies exclusively on the current token x_i, without regard for context. The second component involves the head paths, where the term $(A^h \otimes W_{OV}^h)$ introduces positional mixing. The output at position i becomes a weighted sum over source positions j:

$$\text{Logits}_{\text{head } h} = \sum_{j \leq i} A_{ij}^h \cdot (W_{OV}^h \cdot x_j)$$

In this formulation, A_{ij}^h represents the source selection mechanism (driven by the QK circuit), while $W_{OV}^h x_j$ governs the information copied from that source (driven by the OV circuit). Combining these components, the total logit is the sum:

$$\text{Logits} = \text{Bigram}(x_i) + \sum_h \sum_j \text{Skip-Trigram}(x_j \rightarrow x_i)$$

Thus, the model structurally functions as an ensemble of these heuristics. □

Searching for large entries in the joint W_{QK} and W_{OV} matrices reveals interpretable behavior, such as copying or simple grammatical completion, as shown in Table 1.

Source Token	**Destination Token**	**Out Token**	**Example Skip Tri-grams**
“perfect”	“are”, “looks”, “is”	“perfect”, “super”, “absolute”	“perfect... are perfect”, “perfect... looks super”
“large”	“contains”, “using”, “specify”	“large”, “small”,“very”, “huge”	“large... using large”, “large... contains small”
“two”	“One”,“\n”, “has”	“two”, “three”, “four”, “five”	“two... One two”, “two... has three”
“lambda”	“$\\”, “}{\\”, “+\\”	“lambda”, “sorted”, “operator”	“lambda... $\\lambda”, “lambda... +\\lambda”
“nbsp”	“&”, “\&”, “}&”	“nbsp”, “01”, “gt”, “nbs”	“nbsp... &nbsp”, “nbsp... >&nbsp”
“Great”	“The”, “The”, “the”	“Great”, “great”, “poor”	“Great... The Great”, “Great... the great”

Table 1: Examples of Skip-Trigrams found in One-Layer Transformers. The QK circuit identifies the Source given the Destination; the OV circuit promotes the Out token given the Source. Reproduced from Elhage et al., 2021 [4].

6 Beyond a Single Layer

When we extend the model to two or more layers, the elegance of the independent “Skip-Trigram” paths gives way to significantly higher complexity.

In a one-layer model, every head attends directly to the raw input embeddings. In a two-layer model, however, the heads in Layer 2 attend to a residual stream that contains the *outputs* of Layer 1. In our Kronecker formulation, this corresponds to the multiplication of virtual weights. For example, a Layer 2 head reading from a Layer 1 head results in terms involving $(A^{(2)} \otimes W_{OV}^{(2)})(A^{(1)} \otimes W_{OV}^{(1)})$, effectively chaining the ’Skip-Trigrams’ into longer dependencies.

This leads to an explosion of possible paths. A second-layer head can query information that was moved by a first-layer head, creating “Head-to-Head” circuits. However, rigorously formalizing these composition terms requires expanding the Kronecker product formulation into exponentially many path terms. As such, a full treatment of multi-layer circuits is beyond the scope of this work. We refer the interested reader to a more detailed analysis in Elhage et al. [4].

While our simplified formulations above give a clear and compact representation of attention phenomena, it is important to recognize their limitations. Even with one or two layers, attention-only transformers are already significantly simpler than the full models used in practice.

In a standard transformer, multiple layers of attention interact with large feedforward networks (MLPs) at each layer[3], and residual connections carry information across the network in complex ways. Adding these components back in introduces nonlinearities and interactions that make the behavior of individual neurons and attention heads much harder to interpret.

Nonetheless, studying the linear, attention-only case highlights the core mechanisms by which information flows through attention and how heads interact with embeddings. As we reintroduce MLPs, we encounter phenomena such as *superposition*, where multiple features are represented simultaneously within the same neurons or vector dimensions, further complicating interpretability. Understanding these effects is a key step toward analyzing and reverse-engineering the behavior of larger, more realistic transformers.

[3]For instance, a modern transformer like GPT-3 has 96 such layers, alternating between attention blocks and MLP blocks.

7 Superposition

To discuss superposition, we must first formalize the notion of neurons, neural networks, and multilayer perceptrons (MLPs). While in Definition 5 we introduced transformer MLPs as two-layer position-wise feedforward blocks, here we adopt a more general view of an MLP as a sequence of fully connected layers. A *neuron* is a scalar-valued function $n : \mathbb{R}^d \to \mathbb{R}$ of the form

$$n(x) = \sigma(w^\top x + b),$$

where $w \in \mathbb{R}^d$ is the neuron's weight vector, $b \in \mathbb{R}$ is a bias term, and σ is a monotonically increasing nonlinear map. The weight vector w defines a direction in the representation space, and $n(x)$ measures how strongly the input x aligns with that direction.

A *neural layer* maps an input vector $x \in \mathbb{R}^d$ to an output vector $h \in \mathbb{R}^k$ by applying many such neurons in parallel:

$$h = \sigma(Wx + b),$$

where $W \in \mathbb{R}^{k\times d}$ is the weight matrix whose rows correspond to individual neurons, $b \in \mathbb{R}^k$ is a bias vector, and σ is performed elementwise. Each row of W defines a direction in the layer's representation space, and the layer's output describes how strongly the input aligns with each of these directions.

In deeper networks, layers compose these transformations to build increasingly abstract *features*, which, for our purposes, can be considered as specific directions or subspaces in the activation space that encode meaningful patterns. In large language models, features can align with interpretable semantic directions, such as sentiment or gender associations.

The idea of the *superposition hypothesis* is that neural networks often represent more features than they have neurons. In other words, multiple features can be encoded simultaneously within the same neurons or vector dimensions, allowing the network to efficiently use its limited representational capacity [3].

One reason this strategy is plausible comes from the geometry of high-dimensional spaces. In an n-dimensional space, we can only have n vectors that are perfectly orthogonal. However, as the dimensionality grows, it becomes possible to arrange exponentially many vectors that are *almost orthogonal*. Before formalizing the high-dimensional geometry that permits this, we present a concrete toy model demonstrating how $N = 5$ features can be compressed into $d = 2$ neurons and recovered with minimal error.

Example 26 (The Pentagon Superposition Network)**.** *This example illustrates the core mechanism of superposition: by mapping features to a set of almost orthogonal directions, we enable the approximate recovery of the original vectors via the transpose operation. We construct a toy neural network that compresses* 5 *distinct features into a* 2*-dimensional hidden space. The network is given by the function* $\hat{x} : \mathbb{R}^5 \to \mathbb{R}^5$ *defined as*

$$\hat{x}(x) = \mathrm{ReLU}\big(W_2\, \mathrm{ReLU}(W_1 x + b_1) + b_2\big).$$

Let $e_1, \ldots, e_5$ *denote the standard basis vectors in* $\mathbb{R}^5$*. We associate each feature with a unit vector* $v_k \in \mathbb{R}^2$ *pointing to a vertex of a regular pentagon:*

$$v_k = \left(\cos\frac{2\pi k}{5},\ \sin\frac{2\pi k}{5}\right), \quad k = 1, \ldots, 5.$$

The encoder weights W_1 *are set to the columns of these pentagon vectors, and the encoder bias* b_1 *is chosen to ensure strictly positive hidden activations for basis inputs* ($v_k + b_1 > 0$ *for any* v_k)*:*

$$W_1 = [v_1 \mid \cdots \mid v_5] \in \mathbb{R}^{2\times 5}, \quad b_1 = \begin{bmatrix} 0.82 \\ 0.97 \end{bmatrix}.$$

Now, we define W_2 *and the output bias* b_2 *using a sparsity threshold* $\beta = 0.5$*:*

$$W_2 = 1.5\, W_1^\top, \quad b_2 = -W_2 b_1 - \beta \mathbf{1}_5.$$

First, consider the network's behavior on a single sparse feature $x = e_k$. The hidden activation is simply the shifted embedding vector, $h_k = v_k + b_1$, which is positive entrywise. The pre-activation output $z = W_2 h_k + b_2$ then simplifies to

$$z = W_2(v_k + b_1) - W_2 b_1 - \beta \mathbf{1}_5 = 1.5 W_1^\top v_k - 0.5(\mathbf{1}_5).$$

The j-th component of this output vector is given by the inner product

$$(z)_j = 1.5\langle v_j, v_k \rangle - 0.5.$$

For the target feature ($j = k$), we have $\langle v_k, v_k \rangle = 1$, yielding $(z)_k = 1.5(1) - 0.5 = 1$. For any interfering feature ($j \neq k$), the maximum inner product is $\cos(72^\circ) \approx 0.309$. Consequently,

$$(z)_j \leq 1.5(0.309) - 0.5 \approx -0.04.$$

Since the interference terms are negative, the final ReLU suppresses them completely, resulting in exact conservation of the input: $\hat{x}(e_k) = e_k$. However, this compression comes at a cost. Consider a dense input $x = e_1 + e_3$. Assuming the bias b_1 remains sufficient to keep the hidden state linear, the pre-activation output for the first feature becomes

$$(z)_1 = 1.5\big(\langle v_1, v_1 \rangle + \langle v_1, v_3 \rangle\big) - 0.5.$$

Since v_1 and v_3 are separated by 144°, their dot product is ≈ -0.809. This leads to destructive interference:

$$(z)_1 \approx 1.5(1 - 0.809) - 0.5 \approx -0.21.$$

The final ReLU outputs 0 for this component. Thus, while the network can perfectly store any single feature in the pentagon configuration, attempting to superimpose e_1 and e_3 results in feature erasure due to the "bottleneck".

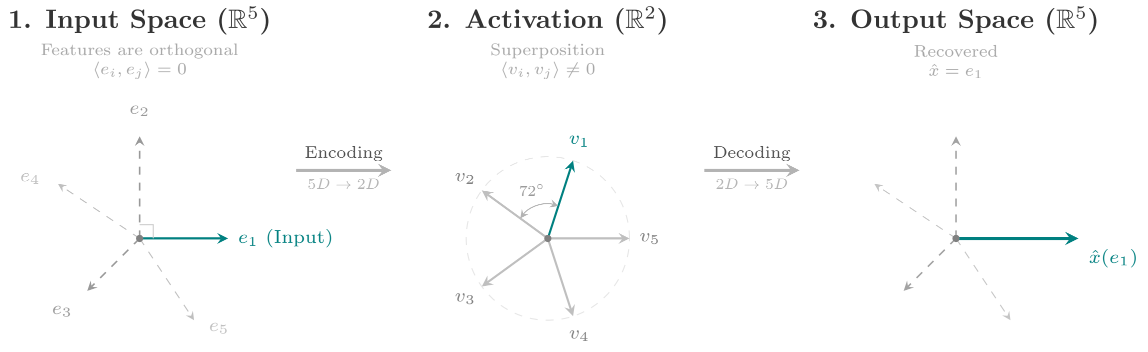

Figure 1: Geometric Superposition Pipeline. The orthogonal input feature e_1 is compressed into a 2D subspace as vector v_1, where it interferes with other features, and is subsequently decoded into the output $\hat{x} = e_1$.

While the toy model relies on specific geometric arrangements (regular polygons), the phenomenon generalizes to high dimensions via the geometry of random projections. In fact, the following result from Johnson-Lindenstrauss (JL) guarantees that a cloud of points in a high-dimensional space can be projected into a much lower-dimensional space while approximately preserving distances (and therefore angles/correlations, see Corollary 28). Additionally, it is proven using *random* projections, suggesting that superposition does not require handcrafted algebraic constructions but can instead emerge from standard random weight matrices used in neural network initialization.

Theorem 27 (Johnson–Lindenstrauss)**.** *Let $0 < \varepsilon < 1$ and let Z be a set of N points in $\mathbb{R}^n$. Then there exists a linear map $f : \mathbb{R}^n \to \mathbb{R}^m$ with*

$$m = O\Big(\frac{\log N}{\varepsilon^2}\Big)$$

such that for all $x, y \in Z$,

$$(1-\varepsilon)\|x-y\|_2^2 \le \|f(x)-f(y)\|_2^2 \le (1+\varepsilon)\|x-y\|_2^2.$$

Proof. Let $Z = \{z^1, \dots, z^N\} \subset \mathbb{R}^n$ be the given set of points. Fix $0 < \varepsilon < 1$. Choose independent random vectors $x^1, \dots, x^m \in \mathbb{R}^n$ by letting each coordinate

$$x_\ell^k \in \left\{ \sqrt{\frac{1+\varepsilon}{m}},\ -\sqrt{\frac{1+\varepsilon}{m}} \right\}$$

with probability 1/2 each. Define a linear map $f : \mathbb{R}^n \to \mathbb{R}^m$ by

$$f(z) = \big(z \cdot x^1,\ z \cdot x^2,\ \dots,\ z \cdot x^m\big).$$

Fix two distinct points $z^i, z^j \in Z$ and set

$$z := z^i - z^j, \qquad u := f(z^i) - f(z^j) = f(z).$$

Then

$$\|u\|_2^2 = \sum_{k=1}^m (x^k \cdot z)^2.$$

For each $k = 1, \dots, m$, define the random variable

$$X_k := (x^k \cdot z)^2.$$

Since the coordinates of x^k are independent, mean zero, and satisfy

$$\mathbb{E}[(x_\ell^k)^2] = \frac{1+\varepsilon}{m},$$

we get

$$\mathbb{E}[X_k] = \sum_{\ell=1}^n z_\ell^2\, \mathbb{E}[(x_\ell^k)^2] = \frac{1+\varepsilon}{m}\, \|z\|_2^2.$$

Thus,

$$\mathbb{E}[\|u\|_2^2] = \sum_{k=1}^m \mathbb{E}[X_k] = (1+\varepsilon)\|z\|_2^2.$$

Each X_k depends on a sum of independent $\pm$ variables and is therefore sufficiently concentrated. In particular, there exist constants $c_1, c_2 > 0$ such that for any $0 < \beta < 1$,

$$\Pr\left[\|u\|_2^2 > (1+\beta)(1+\varepsilon)\|z\|_2^2\right] < e^{-c_1\beta^2 m},$$

$$\Pr\left[\|u\|_2^2 < (1-\beta)(1+\varepsilon)\|z\|_2^2\right] < e^{-c_2\beta^2 m}.$$

Therefore, for some constant $c > 0$,

$$\Pr\Big[\big|\|u\|_2^2 - (1+\varepsilon)\|z\|_2^2\big| > \beta(1+\varepsilon)\|z\|_2^2\Big] < e^{-c\beta^2 m}.$$

There are $\binom{N}{2}$ pairs (i, j). Choose $\beta = \varepsilon/2$. By the union bound, the probability that *any* pair has its squared distance distorted by more than a factor $1 \pm \beta$ is at most

$$\binom{N}{2} e^{-c\varepsilon^2 m/4} \le N^2 e^{-c'\varepsilon^2 m}$$

(for a possibly different constant $c' > 0$). Choose

$$m \geq \frac{C \log N}{\varepsilon^2}$$

with C large enough. Then the above probability is < 1, so there exists a choice of $x^1, \ldots, x^m$ such that *all* pairs satisfy

$$(1 - \tfrac{\varepsilon}{2})(1+\varepsilon)\|z^i - z^j\|_2^2 \;\leq\; \|f(z^i) - f(z^j)\|_2^2 \;\leq\; (1 + \tfrac{\varepsilon}{2})(1+\varepsilon)\|z^i - z^j\|_2^2.$$

For $0 < \varepsilon < 1$,

$$(1 - \tfrac{\varepsilon}{2})(1+\varepsilon) \geq 1, \qquad (1 + \tfrac{\varepsilon}{2})(1+\varepsilon) \leq (1+\varepsilon)^2.$$

Thus,

$$\|z^i - z^j\|_2^2 \leq \|f(z^i) - f(z^j)\|_2^2 \leq (1+\varepsilon)^2 \|z^i - z^j\|_2^2.$$

Taking square roots:

$$\|z^i - z^j\|_2 \leq \|f(z^i) - f(z^j)\|_2 \leq (1+\varepsilon)\|z^i - z^j\|_2.$$

This proves the Johnson–Lindenstrauss lemma for

$$m = O\left(\frac{\log N}{\varepsilon^2}\right).$$

□

Corollary 28. *Let $Z \subset \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^m$ be as in the Johnson–Lindenstrauss (JL) theorem. Then for any two distinct nonzero points $x, y \in Z$, let θ be the angle between x and y, and θ_f the angle between $f(x)$ and $f(y)$. Then*

$$\left|\cos\theta_f - \cos\theta\right| \;\leq\; \frac{4\varepsilon}{1-\varepsilon}.$$

In particular, if x and y are unit vectors,

$$\left|\langle f(x), f(y)\rangle - \langle x, y\rangle\right| \leq 3\varepsilon.$$

Proof. Assume x and y are unit vectors; the general case follows by normalizing. By the polarization identity,

$$\langle x, y\rangle = \frac{\|x\|_2^2 + \|y\|_2^2 - \|x-y\|_2^2}{2}, \qquad \langle f(x), f(y)\rangle = \frac{\|f(x)\|_2^2 + \|f(y)\|_2^2 - \|f(x) - f(y)\|_2^2}{2}.$$

Since JL preserves pairwise distances:

$$|\|f(x) - f(y)\|_2^2 - \|x-y\|_2^2| \leq \varepsilon\|x-y\|_2^2 \leq 4\varepsilon$$

and similarly for $\|x\|^2$ and $\|y\|^2$. Therefore,

$$|\langle f(x), f(y)\rangle - \langle x, y\rangle| \leq 3\varepsilon.$$

Finally, using the fact that $\cos\theta_f = \frac{\langle f(x), f(y)\rangle}{\|f(x)\|\|f(y)\|}$ and $\|f(x)\|, \|f(y)\| \in [\sqrt{1-\varepsilon}, \sqrt{1+\varepsilon}]$, we get

$$|\cos\theta_f - \cos\theta| \leq \frac{4\varepsilon}{1-\varepsilon}.$$

□

8 Conclusion

In this project, we have introduced the Transformer architecture and subsequently simplified it to reveal interpretable, linear algebraic structure that govern how information flows through the network. Without the additional architectural complexities used in practice, such as layer normalization and feed-forward networks, we have shown that the attention mechanism can be compactly represented using the Kronecker product. This notation not only simplifies the algebraic formulation, but also reveals the separable nature of the transformer's operations.

Using this framework, we introduced the "circuit" interpretation of one-layer attention-only transformers. We showed that instead of being considered "black boxes", they can be considered as an ensemble of Bigram and Skip-Trigram heuristics. The Query-Key and Output-Value circuits allow us to read off the model's behavior directly from its weights, identifying how specific tokens attend to and influence one another. Crucially, however, this interpretability encounters significant issues as model depth increases, leading to a combinatorial explosion of such "circuit" paths.

Additionally, the introduction of MLPs result in the phenomenon of superposition, which allows the network to compress features into low-dimensional subspaces, further obscuring interpretability. While the linear algebraic tools presented here provide a solid foundation, understanding the emergent complexity of deeper, non-linear transformers remains a critical task. Continued research is essential not only for optimizing model performance but for ensuring the transparency and safety of the powerful AI systems increasingly integrated into our society.

References

[1] L. Bereska and E. Gavves. Mechanistic interpretability for ai safety – a review, 2024.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[3] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

[4] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, 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. https://transformer-circuits.pub/2021/framework/index.html.

[5] A. Graham. *Kronecker products and matrix calculus*. Mathematics and its Applications. Ellis Horwood Ltd, Publisher, Harlow, England, Nov. 1981.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.