Walking Through a Tiny Transformer

Zach Virgilio

November 7, 2025

The data was a synthetic dataset consisting of the pattern $'ABC'$ repeating over and over. The model architecture is visualized in the following diagram.

$$\begin{tikzcd}
& & \text{softmax}(Q \cdot K^T) \cdot V \arrow[ddr, bend left, "\text{Project Back}"] & & \\
\text{Weight Embedding} \arrow[dr] & & & & \\
& + \arrow[r] \arrow[uur, bend left, "\text{Project Out}"] & Id \arrow[r] & + \arrow[r, "\text{Linear Output}"] & \text{Logits} \\
\text{Position Embedding} \arrow[ur] & & & &
\end{tikzcd}$$

This model only needs to learn 3 facts:

- If I see 'A' as the last term, generate 'B'
- If I see 'B' as the last term, generate 'C'
- If I see 'C' as the last term, generate 'A'

We will walk through the linear algebra of the simple one layer transformer to see how the model encoded these facts.

With a fixed seed of 1234, the model learned a cross-entopry error of 0.0001. We will use it's parameters to manually follow an execution with input ABC. The foward pass naturally predicts a value after each token in context, so our model should predict B follows the first A, C follows the B, and finally, the actual new token prediction, A comes after the last C.

Ultimately, the logits (log probabilities) for a given context are determined by the linear ouput, which is just a linear transformation from the embedding dimension to the vocabulary size, so in this case, it is a 3×2 matrix. In our training run, it ended up being:

$$A_{out} = \begin{bmatrix} -0.5150 & 1.2707 \\ 1.1767 & 1.2045 \\ -0.3778 & -0.6578 \end{bmatrix}$$

When we project to the logits, we will have our vector x passing through the model, and $A_{out}x$ will be:

$$\begin{bmatrix} \begin{bmatrix} -0.5150 & 1.2707 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} \\ \begin{bmatrix} 1.1767 & 1.2405 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} \\ \begin{bmatrix} -0.3778 & -0.6578 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} \end{bmatrix}$$

When the x-component of this vector is largest, the model will most likely predict A. Similarly for y and z, that component being large increases that chance of B and C, respectively. A dot product is maximized when the vector point in the same direction.

- A vector whose components are (roughly) $\begin{bmatrix} -t & 2.5t \end{bmatrix}$ for some positive t will correspond to a prediction that will most likely be A.
- A vector whose component are (roughly) $\begin{bmatrix} t & t \end{bmatrix}$ for some positive t will correspond to the highest likliehood of B.

- A vector component are (roughly) $\begin{bmatrix}-t & 1.75t\end{bmatrix}$ for some positive t will correspond to the highest likliehood of C.

We will pay attention to what operations push vectors to point in these direction to understand how the model has learned to predict the next token in the sequence.

The embedding weights are:

$$A \sim 0 \to W_A = \begin{bmatrix}1.1988\\1.3633\end{bmatrix}, \quad B \sim 1 \to W_B = \begin{bmatrix}-1.8032\\-0.7401\end{bmatrix}, \quad C \sim 2 \to W_C = \begin{bmatrix}-1.0425\\1.4757\end{bmatrix}$$

Where A, B and C are encoded numerically as 0, 1 and 2 respectively. The context size is 3 pieces of information, with position weights:

$$\text{pos } 0 \to W_0 = \begin{bmatrix}-0.3160\\0.9374\end{bmatrix}, \quad \text{pos } 1 \to W_1 = \begin{bmatrix}-0.4459\\1.2218\end{bmatrix}, \quad \text{pos } 2 \to W_2 = \begin{bmatrix}-0.2666\\0.9128\end{bmatrix}$$

The embedding layer for context ABC adds the embedding values of the characters to the embedding values of their respective positions. We thus have the vectors:

$$C_0 = W_A + W_0 = \begin{bmatrix}0.8829\\2.3007\end{bmatrix}, \quad C_1 = W_B + W_1 = \begin{bmatrix}-2.2491\\0.4816\end{bmatrix}, \quad C_2 = W_C + W_1 = \begin{bmatrix}-1.3090\\2.3885\end{bmatrix}$$

Next we apply the single-headed attention to the model. In the model, we have a 6×2 matrix of A weights and 6-dimensional vector of biases b. For computational efficiency, we stack C_0, C_1 and C_2 into a 3×2 matrix where each row is the 2-dimensional vector. The computation is then:

$$\begin{bmatrix}C_0\\C_1\\C_2\end{bmatrix} \cdot A^T + b$$

Where A^T is 2×6, so the dimension computations end up as $(3 \times 2) \cdot (2 \times 6) = 3 \times 6$ and then b is replicated 3 times and added to each row of the resulting 3×6 matrix. However, what is really happening is that A contains 3 different projections. Each C_i must be projected into the transformer by Q, K and V, it's query's, keys and values. So in fact, we decompose A into block matrix form as:

$$A^T = \begin{bmatrix}Q & K & V\end{bmatrix}$$

Call the matrix with rows C_i the context matrix C. We end up computing:

$$\begin{bmatrix}CQ & CK & CV\end{bmatrix}$$

and we split these into 3 separate 3×2 matrices. In general, these would be matrices of size $(\textit{Context size}) \times (\textit{Embedding dimensions})$.