The Mathematics of Transformers: The Architecture Behind LLMs

UNIVERSITY OF OXFORD

Mathematical Institute

LIDA KANARI

Mathematical Institute
University of Oxford

Introduction to Machine Learning, November 2025

Oxford Mathematics

Table of Contents

What is a Large Language Model (LLM)?

Definition. A Large Language Model (LLM) is a transformer-based neural network trained on large text to predict the next token in a sequence.

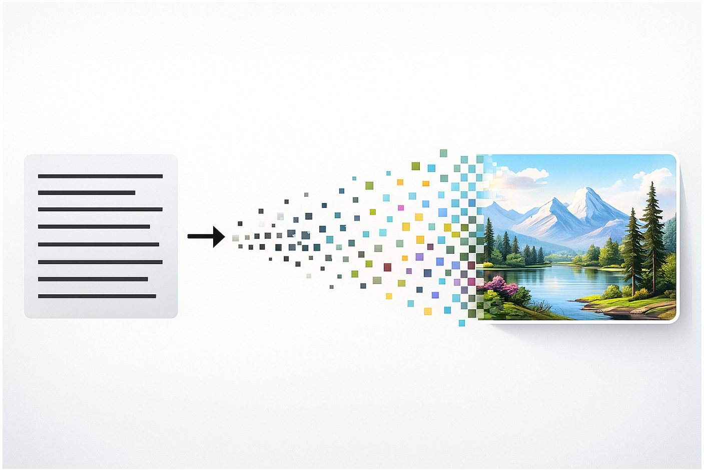
$$P(w_t \mid w_1, \ldots, w_{t-1})$$

It learns language structure and meaning by optimizing this conditional probability.

Transformer Applications

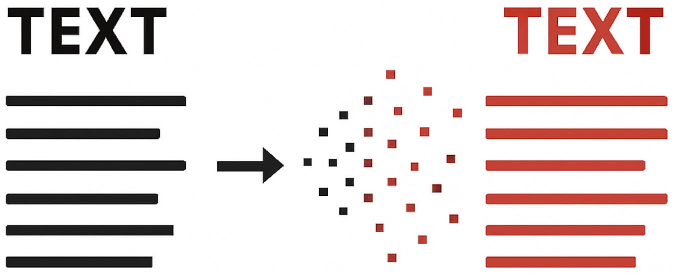
- **Audio $\rightarrow$ Text:** speech recognition (*Whisper*)
- **Text $\rightarrow$ Image:** generative models (*DALL·E*)
- **Translation:** sequence-to-sequence models (*T5*)
- **Chat / Completion:** GPT family of models

Transformer Applications



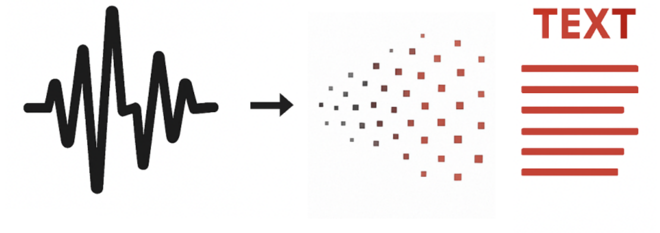
Transformer: Text to image.

Transformer Applications



Transformer: Text to text.

Transformer Applications



Transformer: Sound to text.

Key Elements of a Transformer

- **Embedding:** map tokens to vectors.
- **Query (Q), Key (K), Value (V):** derive attention weights.
- **Attention output:** context-aware representation.
- **MultiLayer Perceptron (MLP):** encodes memories.
- **Unembedding:** map back to vocabulary logits.

Predicting the Next Word

Goal. Given a partial sentence, predict the most likely next word.

$$P(w_t \mid w_1, \dots, w_{t-1})$$

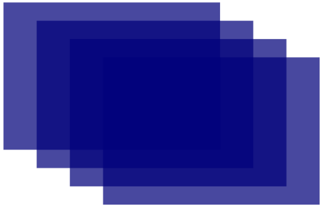
Example: "The quick brown fox jumps over the lazy dog ___" $\to$ model predicts "dog".

Tokenizing a Sentence

The quick brown fox jumps over the lazy dog.

$\boxed{\text{The}}$ $\boxed{\text{quick}}$ $\boxed{\text{brown}}$ $\boxed{\text{fox}}$ $\boxed{\text{jumps}}$ $\boxed{\text{over}}$ $\boxed{\text{the}}$ $\boxed{\text{lazy}}$ $\boxed{\text{dog.}}$

Next-word Probability prediction



Transformer

$\longrightarrow$

The next word in the sentence is ...

predicted	0.42
from a	0.32
sequence	0.28
LLM	0.15
model	0.10
output	0.06
sampled	0.04

Embedding of words

Tokenizing and Embedding a Sentence

The quick brown fox jumps over the lazy dog.

$$\begin{tikzcd}[column sep=small]
\boxed{\text{The}} \arrow[d] & \boxed{\text{quick}} \arrow[d] & \boxed{\text{brown}} \arrow[d] & \boxed{\text{fox}} \arrow[d] & \boxed{\text{jumps}} \arrow[d] & \boxed{\text{over}} \arrow[d] & \boxed{\text{the}} \arrow[d] & \boxed{\text{lazy}} \arrow[d] & \boxed{\text{dog.}} \arrow[d] \\
\begin{bmatrix} 0.42 \\ -0.88 \\ 0.13 \\ 0.77 \\ -0.55 \end{bmatrix} & \begin{bmatrix} -0.31 \\ 0.92 \\ -0.44 \\ -0.05 \\ 0.61 \end{bmatrix} & \begin{bmatrix} 0.73 \\ -0.22 \\ 0.48 \\ -0.66 \\ 0.19 \end{bmatrix} & \begin{bmatrix} -0.57 \\ 0.11 \\ -0.94 \\ 0.28 \\ 0.36 \end{bmatrix} & \begin{bmatrix} 0.15 \\ 0.84 \\ -0.07 \\ -0.72 \\ 0.53 \end{bmatrix} & \begin{bmatrix} -0.62 \\ 0.39 \\ -0.12 \\ 0.91 \\ -0.27 \end{bmatrix} & \begin{bmatrix} 0.58 \\ -0.49 \\ 0.33 \\ 0.12 \\ -0.81 \end{bmatrix} & \begin{bmatrix} -0.04 \\ 0.67 \\ -0.73 \\ 0.45 \\ 0.29 \end{bmatrix} & \begin{bmatrix} 0.91 \\ -0.35 \\ 0.26 \\ -0.58 \\ 0.72 \end{bmatrix}
\end{tikzcd}$$

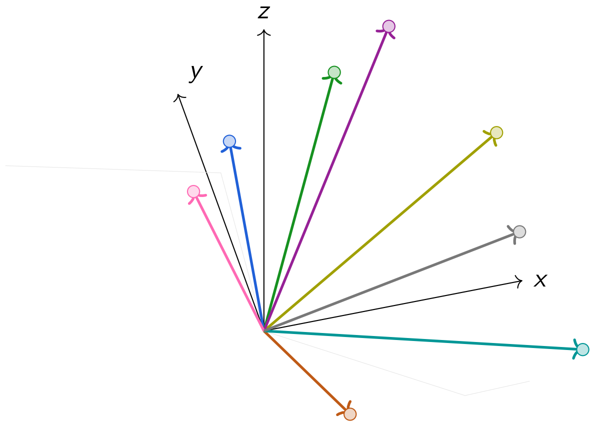
Tokenizing and Embedding a Sentence

The quick brown fox jumps over the lazy dog.

$$\begin{tikzcd}
\boxed{\text{The}} \arrow[d] & \boxed{\text{quick}} \arrow[d] & \boxed{\text{brown}} \arrow[d] & \boxed{\text{fox}} \arrow[d] & \boxed{\text{jumps}} \arrow[d] & \boxed{\text{over}} \arrow[d] & \boxed{\text{the}} \arrow[d] & \boxed{\text{lazy}} \arrow[d] & \boxed{\text{dog.}} \arrow[d] \\
v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9
\end{tikzcd}$$

Illustrative 3D Sketch of Token Vectors

3D view of the token vectors



Vocabulary and Token Indices

The vocabulary is a predefined list of tokens (words or sub-words). For example for GPT-3, the vocabulary is of size ≈ 50000.

$$\text{Vocabulary: } \mathcal{V} = \{w_1, \ldots, w_{50000}\}$$

Each token w_i is assigned an integer ID i.

Embedding: Mapping Tokens to Vectors

Each token index i is mapped to a vector $E_i \in \mathbb{R}^d$:

$$E \in \mathbb{R}^{d \times \mathcal{V}},$$

where d is the embedding dimension.

Semantic Structure in Embeddings

During training, the model discovers an embedding space where directions have semantic meaning. Example:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}.$$

Such vector arithmetic encodes linguistic relationships.

Direction Matters

The direction between male/female words is similar to that between uncle/aunt.

$$v_{\text{male} \rightarrow \text{female}} \approx v_{\text{uncle} \rightarrow \text{aunt}}$$

This property arises naturally from training objectives.

Dot Product is Measuring Similarity

For two embedding vectors $u, v \in \mathbb{R}^d$:

$$u \cdot v = \sum_{i=1}^{d} u_i v_i$$

Interpretation:
large $\rightarrow$ vectors aligned (similar meaning)
small $\rightarrow$ unrelated.

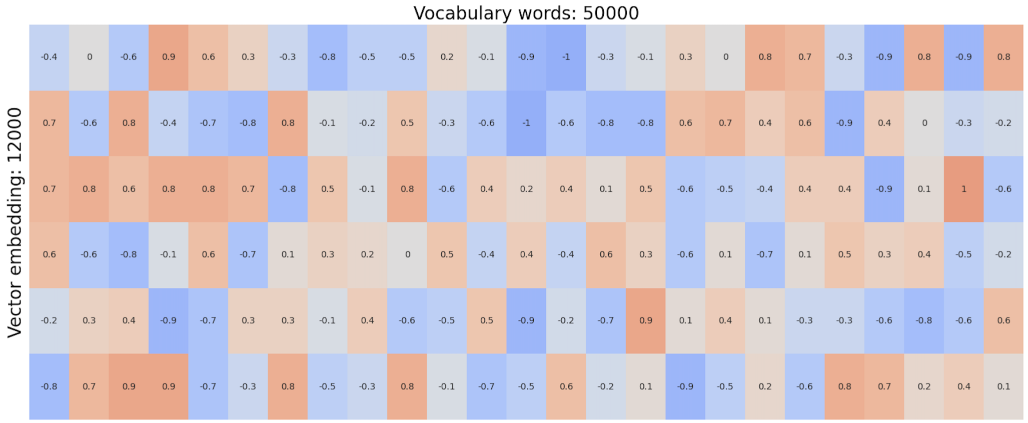
Vocabulary Size in GPT-3

GPT-3 uses a byte-pair encoding (BPE) vocabulary of about 50 000 tokens.

$$|\mathcal{V}| \approx 5 \times 10^4$$

Each token has a learned embedding vector of dimension $d = 12288$.

Embedding matrix



-0.4	0	-0.6	0.9	0.6	0.3	-0.3	-0.8	-0.5	-0.5	0.2	-0.1	-0.9	-1	-0.3	-0.1	0.3	0	0.8	0.7	-0.3	-0.9	0.8	-0.9	0.8
0.7	-0.6	0.8	-0.4	-0.7	-0.8	0.8	-0.1	-0.2	0.5	-0.3	-0.6	-1	-0.6	-0.8	-0.8	0.6	0.7	0.4	0.6	-0.9	0.4	0	-0.3	-0.2
0.7	0.8	0.6	0.8	0.8	0.7	-0.8	0.5	-0.1	0.8	-0.6	0.4	0.2	0.4	0.1	0.5	-0.6	-0.5	-0.4	0.4	0.4	-0.9	0.1	1	-0.6
0.6	-0.6	-0.8	-0.1	0.6	-0.7	0.1	0.3	0.2	0	0.5	-0.4	0.4	-0.4	0.6	0.3	-0.6	0.1	-0.7	0.1	0.5	0.3	0.4	-0.5	-0.2
-0.2	0.3	0.4	-0.9	-0.7	0.3	0.3	-0.1	0.4	-0.6	-0.5	0.5	-0.9	-0.2	-0.7	0.9	0.1	0.4	0.1	-0.3	-0.3	-0.6	-0.8	-0.6	0.6
-0.8	0.7	0.9	0.9	-0.7	-0.3	0.8	-0.5	-0.3	0.8	-0.1	-0.7	-0.5	0.6	-0.2	0.1	-0.9	-0.5	0.2	-0.6	0.8	0.7	0.2	0.4	0.1

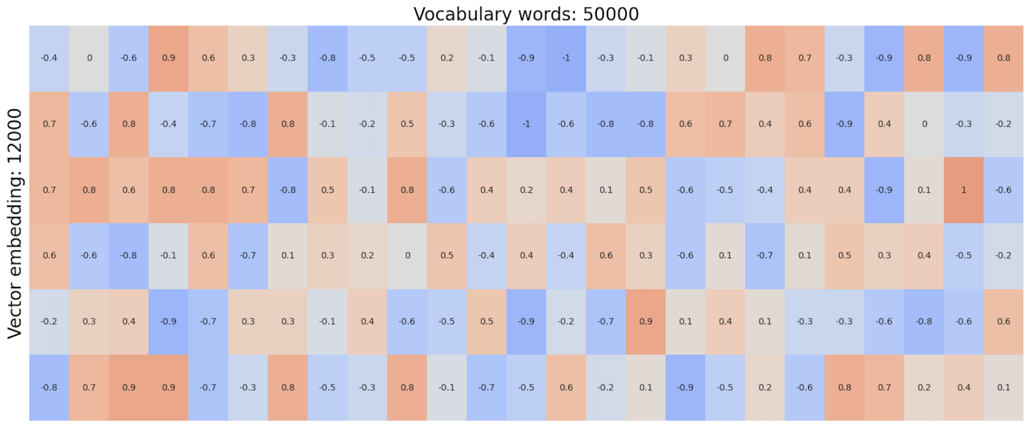
Embedding matrix

Embedding matrix parameters

- **Dimensions:**
  vocabulary size $\mathcal{V} \approx 50{,}000$,
  embedding dimension $d \approx 12{,}000$.
- **Total parameters:**
  $P_{emb} = \mathcal{V} \times d \approx 600{,}000{,}000$ parameters.

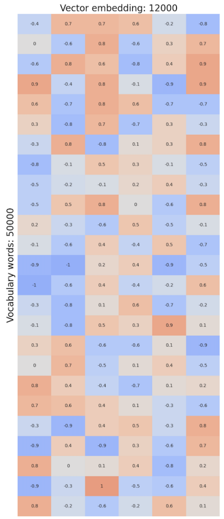
Un-embedding of words

Embedding matrix



-0.4	0	-0.6	0.9	0.6	0.3	-0.3	-0.8	-0.5	-0.5	0.2	-0.1	-0.9	-1	-0.3	-0.1	0.3	0	0.8	0.7	-0.3	-0.9	0.8	-0.9	0.8
0.7	-0.6	0.8	-0.4	-0.7	-0.8	0.8	-0.1	-0.2	0.5	-0.3	-0.6	-1	-0.6	-0.8	-0.8	0.6	0.7	0.4	0.6	-0.9	0.4	0	-0.3	-0.2
0.7	0.8	0.6	0.8	0.8	0.7	-0.8	0.5	-0.1	0.8	-0.6	0.4	0.2	0.4	0.1	0.5	-0.6	-0.5	-0.4	0.4	0.4	-0.9	0.1	1	-0.6
0.6	-0.6	-0.8	-0.1	0.6	-0.7	0.1	0.3	0.2	0	0.5	-0.4	0.4	-0.4	0.6	0.3	-0.6	0.1	-0.7	0.1	0.5	0.3	0.4	-0.5	-0.2
-0.2	0.3	0.4	-0.9	-0.7	0.3	0.3	-0.1	0.4	-0.6	-0.5	0.5	-0.9	-0.2	-0.7	0.9	0.1	0.4	0.1	-0.3	-0.3	-0.6	-0.8	-0.6	0.6
-0.8	0.7	0.9	0.9	-0.7	-0.3	0.8	-0.5	-0.3	0.8	-0.1	-0.7	-0.5	0.6	-0.2	0.1	-0.9	-0.5	0.2	-0.6	0.8	0.7	0.2	0.4	0.1

Un-embedding matrix



-0.4	0.7	0.7	0.6	-0.2	-0.8
0	-0.6	0.8	-0.6	0.3	0.7
-0.6	0.8	0.6	-0.8	0.4	0.9
0.9	-0.4	0.8	-0.1	-0.9	0.9
0.6	-0.7	0.8	0.6	-0.7	-0.7
0.3	-0.8	0.7	-0.7	0.3	-0.3
-0.3	0.8	-0.8	0.1	0.3	0.8
-0.8	-0.1	0.5	0.3	-0.1	-0.5
-0.5	-0.2	-0.1	0.2	0.4	-0.3
-0.5	0.5	0.8	0	-0.6	0.8
0.2	-0.3	-0.6	0.5	-0.5	-0.1
-0.1	-0.6	0.4	-0.4	0.5	-0.7
-0.9	-1	0.2	0.4	-0.9	-0.5
-1	-0.6	0.4	-0.4	-0.2	0.6
-0.3	-0.8	0.1	0.6	-0.7	-0.2
-0.1	-0.8	0.5	0.3	0.9	0.1
0.3	0.6	-0.6	-0.6	0.1	-0.9
0	0.7	-0.5	0.1	0.4	-0.5
0.8	0.4	-0.4	-0.7	0.1	0.2
0.7	0.6	0.4	0.1	-0.3	-0.6
-0.3	-0.9	0.4	0.5	-0.3	0.8
-0.9	0.4	-0.9	0.3	-0.6	0.7
0.8	0	0.1	0.4	-0.8	0.2
-0.9	-0.3	1	-0.5	-0.6	0.4
0.8	-0.2	-0.6	-0.2	0.6	0.1

Un-embedding matrix

Un-embedding matrix parameters

- **Dimensions:**
  vocabulary size $\mathcal{V} \approx 50{,}000$,
  embedding dimension $d \approx 12{,}000$.
- **Total parameters:**
  $P_{unemb} = \mathcal{V} \times d \approx 600{,}000{,}000$ parameters.

Unembedding Matrix W_U

To convert hidden states back to vocabulary logits:

$$z = W_U\, h_t, \quad W_U \in \mathbb{R}^{\mathcal{V} \times d}.$$

Each row of W_U corresponds to a word in the vocabulary. It is approximately the transpose of the embedding matrix.

Softmax transformation

Softmax

Turning arbitrary real scores into a probability distribution

- **Why we need softmax:** model outputs (logits / scores) are real numbers that must be converted into probabilities to make decisions.
- **Desired properties of the output:**
  - each probability is between 0 and 1,
  - the probabilities sum to 1 (a proper probability distribution).
- Softmax guarantees both properties while preserving relative ordering of scores (monotonic with respect to score differences).

Softmax

From scores to probabilities

$$\text{Softmax}(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}} \qquad \text{for } \mathbf{x} = (x_1, \ldots, x_n).$$

Notes:

- Exponentiation makes all outputs positive.
- Division by the sum normalizes them to sum to 1.
- Numerically stable implementation

How softmax works

Flow: scores $\rightarrow$ exp $\rightarrow$ sum $\rightarrow$ normalize

$$\begin{tikzcd}
\boxed{\begin{matrix}\text{scores (logits)}\\ \begin{bmatrix}1.00\\2.00\\0.50\end{bmatrix}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{exponentiate}\\ \begin{bmatrix}e^{x_1}\\e^{x_2}\\e^{x_3}\end{bmatrix}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{sum}\\ S = \sum_i e^{x_i}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{probabilities}\\ \frac{1}{S}\begin{bmatrix}e^{x_1}\\e^{x_2}\\e^{x_3}\end{bmatrix}\end{matrix}}
\end{tikzcd}$$

How softmax works

Input: $\mathbf{x} = (1.00,\ 2.00,\ 0.50)$

Exponentials: $e^{x} \approx \begin{bmatrix} 2.7183 \\ 7.3891 \\ 1.6487 \end{bmatrix}$

Sum: $S \approx 11.7561$

Final probabilities: $\mathrm{softmax}(\mathbf{x}) \approx \begin{bmatrix} 0.2312 \\ 0.6285 \\ 0.1402 \end{bmatrix}$

How softmax works

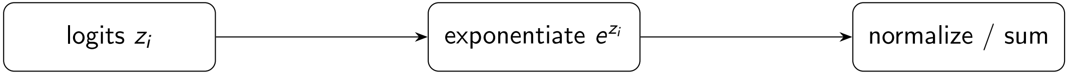
Given logits $z = (z_1, \ldots, z_{|\mathcal{V}|})$, the softmax produces probabilities:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

Properties:

- $p_i > 0$, $\sum_i p_i = 1$.
- Amplifies differences between logits.

Softmax – Schematic



Output: probabilities p_i

Softmax Intuition

- Small logits $\rightarrow$ probabilities 0.
- Large logits $\rightarrow$ probabilities 1 (dominant token).
- Converts scores into a categorical distribution.

Softmax with Temperature T

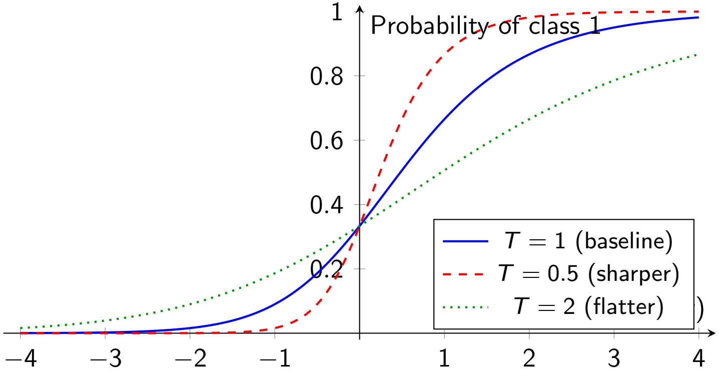
Introduce a temperature $T > 0$ to control output randomness:

$$p_i(T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}.$$

Behaviour:

- $T > 1 \to$ flatter distribution (more random).
- $T < 1 \to$ sharper distribution (more deterministic).

Softmax Temperature Effects



Same logits, different temperatures T in $p_i(T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$.

Softmax Temperature Scaling

- **High** $T \rightarrow$ probabilities spread across many tokens $\rightarrow$ creative but unstable.
- **Low** $T \rightarrow$ probabilities concentrated on a few $\rightarrow$ predictable, repetitive.

Temperature Practical Range

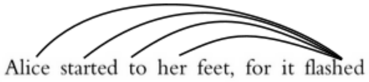
- $T = 0$: output becomes argmax (no randomness).
- Typical range in LLMs: $T \in [0, 2]$.
- Lower T constrains output; higher T increases diversity.

Attention

Attention

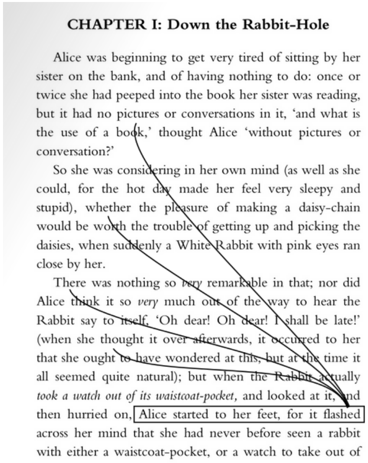
The concept of attention demonstrates how the model integrates information from different parts of the text. It ranges from attention within a sentence to attention in previous parts of the text. Attention is quite expensive, so most models limit the attention context window.

Attention Local - vs - Global



Local attention

Attention Local - vs - Global



CHAPTER I: Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

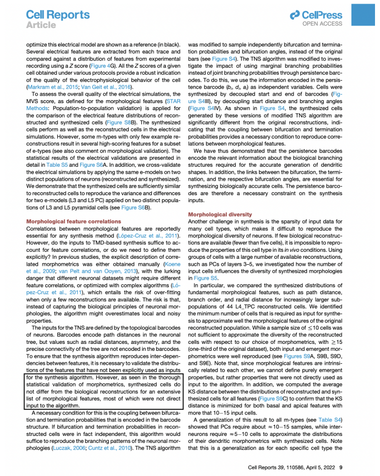
So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so *very* remarkable in that; nor did Alice think it so *very* much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit *actually took a watch out of its waistcoat-pocket,* and looked at it, and then hurried on, $\boxed{\text{Alice started to her feet, for it flashed}}$ across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of

Global attention

Attention Focus

Mathematical Institute



Cell Reports
Article

CellPress
OPEN ACCESS

optimize this electrical model are shown as a reference (in black). Several electrical features are extracted from each trace and compared against a distribution of features from experimental recording using a *Z* score (Figure 4G). All the *Z* scores of a given cell obtained under various protocols provide a robust indication of the quality of the electrophysiological behavior of the cell (Markram et al., 2015; Van Geit et al., 2016).

To assess the overall quality of the electrical simulations, the MVS score, as defined for the morphological features (STAR Methods: Population-to-population validation) is applied for the comparison of the electrical feature distributions of reconstructed and synthesized cells (Figure S8B). The synthesized cells perform as well as the reconstructed cells in the electrical simulations. However, some m-types with only few example reconstructions result in several high-scoring features for a subset of e-types (see also comment on morphological validation). The statistical results of the electrical validations are presented in detail in Table S5 and Figure S6A. In addition, we cross-validate the electrical simulations by applying the same e-models on two distinct populations of neurons (reconstructed and synthesized). We demonstrate that the synthesized cells are sufficiently similar to reconstructed cells to reproduce the variance and differences for two e-models (L3 and L5 PC) applied on two distinct populations of L3 and L5 pyramidal cells (see Figure S6B).

Morphological feature correlations

Correlations between morphological features are reportedly essential for any synthesis method (López-Cruz et al., 2011). However, do the inputs to TMD-based synthesis suffice to account for feature correlations, or do we need to define them explicitly? In previous studies, the explicit description of correlated morphometrics was either obtained manually (Koene et al., 2009; van Pelt and van Ooyen, 2013), with the lurking danger that different neuronal datasets might require different feature correlations, or optimized with complex algorithms (López-Cruz et al., 2011), which entails the risk of over-fitting when only a few reconstructions are available. The risk is that, instead of capturing the biological principles of neuronal morphologies, the algorithm might overestimates local and noisy properties.

The inputs for the TNS are defined by the topological barcodes of neurons. Barcodes encode path distances in the neuronal tree, but values such as radial distances, asymmetry, and the precise connectivity of the tree are not encoded in the barcodes. To ensure that the synthesis algorithm reproduces inter-dependencies between features, it is necessary to validate the distributions of the features that have not been explicitly used as inputs for the synthesis algorithm. However, as seen in the thorough statistical validation of morphometrics, synthesized cells do not differ from the biological reconstructions for an extensive list of morphological features, most of which were not direct input to the algorithm.

A necessary condition for this is the coupling between bifurcation and termination probabilities that is encoded in the barcode structure. If bifurcation and termination probabilities in reconstructed cells were in fact independent, this algorithm would suffice to reproduce the branching patterns of the neuronal morphologies (Luczak, 2006; Cuntz et al., 2010). The TNS algorithm was modified to sample independently bifurcation and termination probabilities and bifurcation angles, instead of the original bars (see Figure S4). The TNS algorithm was modified to investigate the impact of using marginal branching probabilities instead of joint branching probabilities through persistence barcodes. To do this, we use the information encoded in the persistence barcode (b_i, d_i, a_i) as independent variables. Cells were synthesized by decoupled start and end of barcodes (Figure S4III), by decoupling start distance and branching angles (Figure S4IV). As shown in Figure S4, the synthesized cells generated by these versions of modified TNS algorithm are significantly different from the original reconstructions, indicating that the coupling between bifurcation and termination probabilities provides a necessary condition to reproduce correlations between morphological features.

We have thus demonstrated that the persistence barcodes encode the relevant information about the biological branching structures required for the accurate generation of dendritic shapes. In addition, the links between the bifurcation, the termination, and the respective bifurcation angles, are essential for synthesizing biologically accurate cells. The persistence barcodes are therefore a necessary constraint on the synthesis inputs.

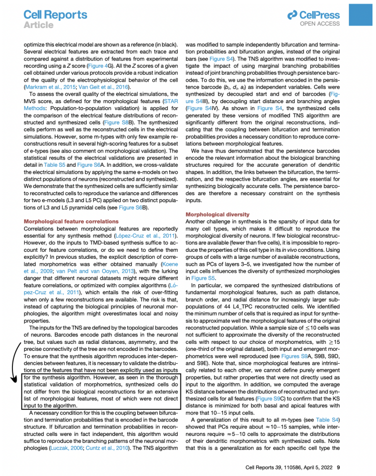
Morphological diversity

Another challenge in synthesis is the sparsity of input data for many cell types, which makes it difficult to reproduce the morphological diversity of neurons. If few biological reconstructions are available (fewer than five cells), it is impossible to reproduce the properties of this cell type in its *in vivo* conditions. Using groups of cells with a large number of available reconstructions, such as PCs of layers 3–5, we investigated how the number of input cells influences the diversity of synthesized morphologies in Figure S5.

In particular, we compared the synthesized distributions of fundamental morphological features, such as path distance, branch order, and radial distance for increasingly larger subpopulations of 44 L4_TPC reconstructed cells. We identified the minimum number of cells that is required as input for synthesis to approximate well the morphological features of the original reconstructed population. While a sample size of $\leq$10 cells was not sufficient to approximate the diversity of the reconstructed cells with respect to our choice of morphometrics, with $\geq$15 (one-third of the original dataset), both input and emergent morphometrics were well reproduced (see Figures S9A, S9B, S9D, and S9E). Note that, since morphological features are intrinsically related to each other, we cannot define purely emergent properties, but rather properties that were not directly used as input to the algorithm. In addition, we computed the average KS distance between the distributions of reconstructed and synthesized cells for all features (Figure S9C) to confirm that the KS distance is minimized for both basal and apical features with more that 10–15 input cells.

A generalization of this result to all m-types (see Table S4) showed that PCs require about $\approx$10–15 samples, while interneurons require $\approx$5–10 cells to approximate the distributions of their dendritic morphometrics with synthesized cells. Note that this is a generalization as for each specific cell type the

Attention Context Window



optimize this electrical model are shown as a reference (in black). Several electrical features are extracted from each trace and compared against a distribution of features from experimental recording using a *Z* score (Figure 4G). All the *Z* scores of a given cell obtained under various protocols provide a robust indication of the quality of the electrophysiological behavior of the cell (Markram et al., 2015; Van Geit et al., 2016).

To assess the overall quality of the electrical simulations, the MVS score, as defined for the morphological features (STAR Methods: Population-to-population validation) is applied for the comparison of the electrical feature distributions of reconstructed and synthesized cells (Figure S8B). The synthesized cells perform as well as the reconstructed cells in the electrical simulations. However, some m-types with only few example reconstructions result in several high-scoring features for a subset of e-types (see also comment on morphological validation). The statistical results of the electrical validations are presented in detail in Table S5 and Figure S6A. In addition, we cross-validate the electrical simulations by applying the same e-models on two distinct populations of neurons (reconstructed and synthesized). We demonstrate that the synthesized cells are sufficiently similar to reconstructed cells to reproduce the variance and differences for two e-models (L3 and L5 PC) applied on two distinct populations of L3 and L5 pyramidal cells (see Figure S6B).

Morphological feature correlations

Correlations between morphological features are reportedly essential for any synthesis method (López-Cruz et al., 2011). However, do the inputs to TMD-based synthesis suffice to account for feature correlations, or do we need to define them explicitly? In previous studies, the explicit description of correlated morphometrics was either obtained manually (Koene et al., 2009; van Pelt and van Ooyen, 2013), with the lurking danger that different neuronal datasets might require different feature correlations, or optimized with complex algorithms (López-Cruz et al., 2011), which entails the risk of over-fitting when only a few reconstructions are available. The risk is that, instead of capturing the biological principles of neuronal morphologies, the algorithm might overestimates local and noisy properties.

The inputs for the TNS are defined by the topological barcodes of neurons. Barcodes encode path distances in the neuronal tree, but values such as radial distances, asymmetry, and the precise connectivity of the tree are not encoded in the barcodes. To ensure that the synthesis algorithm reproduces inter-dependencies between features, it is necessary to validate the distributions of the features that have not been explicitly used as inputs for the synthesis algorithm. However, as seen in the thorough statistical validation of morphometrics, synthesized cells do not differ from the biological reconstructions for an extensive list of morphological features, most of which were not direct input to the algorithm.

A necessary condition for this is the coupling between bifurcation and termination probabilities that is encoded in the barcode structure. If bifurcation and termination probabilities in reconstructed cells were in fact independent, this algorithm would suffice to reproduce the branching patterns of the neuronal morphologies (Luczak, 2006; Cuntz et al., 2010). The TNS algorithm was modified to sample independently bifurcation and termination probabilities and bifurcation angles, instead of the original bars (see Figure S4). The TNS algorithm was modified to investigate the impact of using marginal branching probabilities instead of joint branching probabilities through persistence barcodes. To do this, we use the information encoded in the persistence barcode (b_i, d_i, a_i) as independent variables. Cells were synthesized by decoupled start and end of barcodes (Figure S4III), by decoupling start distance and branching angles (Figure S4IV). As shown in Figure S4, the synthesized cells generated by these versions of modified TNS algorithm are significantly different from the original reconstructions, indicating that the coupling between bifurcation and termination probabilities provides a necessary condition to reproduce correlations between morphological features.

We have thus demonstrated that the persistence barcodes encode the relevant information about the biological branching structures required for the accurate generation of dendritic shapes. In addition, the links between the bifurcation, the termination, and the respective bifurcation angles, are essential for synthesizing biologically accurate cells. The persistence barcodes are therefore a necessary constraint on the synthesis inputs.

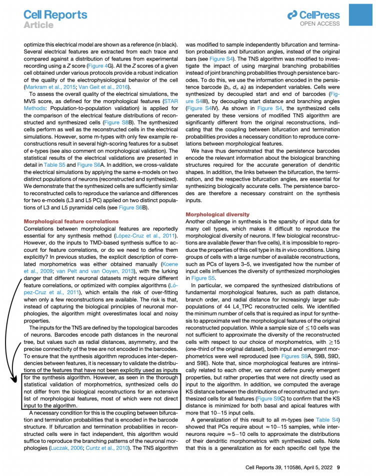
Morphological diversity

Another challenge in synthesis is the sparsity of input data for many cell types, which makes it difficult to reproduce the morphological diversity of neurons. If few biological reconstructions are available (fewer than five cells), it is impossible to reproduce the properties of this cell type in its *in vivo* conditions. Using groups of cells with a large number of available reconstructions, such as PCs of layers 3–5, we investigated how the number of input cells influences the diversity of synthesized morphologies in Figure S5.

In particular, we compared the synthesized distributions of fundamental morphological features, such as path distance, branch order, and radial distance for increasingly larger subpopulations of 44 L4_TPC reconstructed cells. We identified the minimum number of cells that is required as input for synthesis to approximate well the morphological features of the original reconstructed population. While a sample size of $\leq$10 cells was not sufficient to approximate the diversity of the reconstructed cells with respect to our choice of morphometrics, with $\geq$15 (one-third of the original dataset), both input and emergent morphometrics were well reproduced (see Figures S9A, S9B, S9D, and S9E). Note that, since morphological features are intrinsically related to each other, we cannot define purely emergent properties, but rather properties that were not directly used as input to the algorithm. In addition, we computed the average KS distance between the distributions of reconstructed and synthesized cells for all features (Figure S9C) to confirm that the KS distance is minimized for both basal and apical features with more that 10−15 input cells.

A generalization of this result to all m-types (see Table S4) showed that PCs require about ≈10−15 samples, while interneurons require ≈5−10 cells to approximate the distributions of their dendritic morphometrics with synthesized cells. Note that this is a generalization as for each specific cell type the

Attention Context Window



Cell Reports
Article

CellPress
OPEN ACCESS

optimize this electrical model are shown as a reference (in black). Several electrical features are extracted from each trace and compared against a distribution of features from experimental recording using a *Z* score (Figure 4G). All the *Z* scores of a given cell obtained under various protocols provide a robust indication of the quality of the electrophysiological behavior of the cell (Markram et al., 2015; Van Geit et al., 2016).

To assess the overall quality of the electrical simulations, the MVS score, as defined for the morphological features (STAR Methods: Population-to-population validation) is applied for the comparison of the electrical feature distributions of reconstructed and synthesized cells (Figure S8B). The synthesized cells perform as well as the reconstructed cells in the electrical simulations. However, some m-types with only few example reconstructions result in several high-scoring features for a subset of e-types (see also comment on morphological validation). The statistical results of the electrical validations are presented in detail in Table S5 and Figure S6A. In addition, we cross-validate the electrical simulations by applying the same e-models on two distinct populations of neurons (reconstructed and synthesized). We demonstrate that the synthesized cells are sufficiently similar to reconstructed cells to reproduce the variance and differences for two e-models (L3 and L5 PC) applied on two distinct populations of L3 and L5 pyramidal cells (see Figure S6B).

Morphological feature correlations

Correlations between morphological features are reportedly essential for any synthesis method (López-Cruz et al., 2011). However, do the inputs to TMD-based synthesis suffice to account for feature correlations, or do we need to define them explicitly? In previous studies, the explicit description of correlated morphometrics was either obtained manually (Koene et al., 2009; van Pelt and van Ooyen, 2013), with the lurking danger that different neuronal datasets might require different feature correlations, or optimized with complex algorithms (López-Cruz et al., 2011), which entails the risk of over-fitting when only a few reconstructions are available. The risk is that, instead of capturing the biological principles of neuronal morphologies, the algorithm might overestimates local and noisy properties.

The inputs for the TNS are defined by the topological barcodes of neurons. Barcodes encode path distances in the neuronal tree, but values such as radial distances, asymmetry, and the precise connectivity of the tree are not encoded in the barcodes. To ensure that the synthesis algorithm reproduces inter-dependencies between features, it is necessary to validate the distributions of the features that have not been explicitly used as inputs for the synthesis algorithm. However, as seen in the thorough statistical validation of morphometrics, synthesized cells do not differ from the biological reconstructions for an extensive list of morphological features, most of which were not direct input to the algorithm.

A necessary condition for this is the coupling between bifurcation and termination probabilities that is encoded in the barcode structure. If bifurcation and termination probabilities in reconstructed cells were in fact independent, this algorithm would suffice to reproduce the branching patterns of the neuronal morphologies (Luczak, 2006; Cuntz et al., 2010). The TNS algorithm was modified to sample independently bifurcation and termination probabilities and bifurcation angles, instead of the original bars (see Figure S4). The TNS algorithm was modified to investigate the impact of using marginal branching probabilities instead of joint branching probabilities through persistence barcodes. To do this, we use the information encoded in the persistence barcode (b_i, d_i, a_i) as independent variables. Cells were synthesized by decoupled start and end of barcodes (Figure S4III), by decoupling start distance and branching angles (Figure S4IV). As shown in Figure S4, the synthesized cells generated by these versions of modified TNS algorithm are significantly different from the original reconstructions, indicating that the coupling between bifurcation and termination probabilities provides a necessary condition to reproduce correlations between morphological features.

We have thus demonstrated that the persistence barcodes encode the relevant information about the biological branching structures required for the accurate generation of dendritic shapes. In addition, the links between the bifurcation, the termination, and the respective bifurcation angles, are essential for synthesizing biologically accurate cells. The persistence barcodes are therefore a necessary constraint on the synthesis inputs.

Morphological diversity

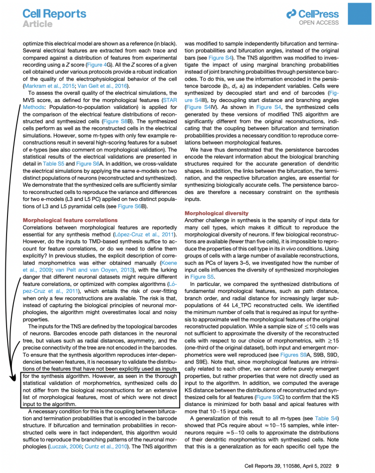
Another challenge in synthesis is the sparsity of input data for many cell types, which makes it difficult to reproduce the morphological diversity of neurons. If few biological reconstructions are available (fewer than five cells), it is impossible to reproduce the properties of this cell type in its *in vivo* conditions. Using groups of cells with a large number of available reconstructions, such as PCs of layers 3–5, we investigated how the number of input cells influences the diversity of synthesized morphologies in Figure S5.

In particular, we compared the synthesized distributions of fundamental morphological features, such as path distance, branch order, and radial distance for increasingly larger subpopulations of 44 L4_TPC reconstructed cells. We identified the minimum number of cells that is required as input for synthesis to approximate well the morphological features of the original reconstructed population. While a sample size of ≤ 10 cells was not sufficient to approximate the diversity of the reconstructed cells with respect to our choice of morphometrics, with ≥ 15 (one-third of the original dataset), both input and emergent morphometrics were well reproduced (see Figures S9A, S9B, S9D, and S9E). Note that, since morphological features are intrinsically related to each other, we cannot define purely emergent properties, but rather properties that were not directly used as input to the algorithm. In addition, we computed the average KS distance between the distributions of reconstructed and synthesized cells for all features (Figure S9C) to confirm that the KS distance is minimized for both basal and apical features with more that 10−15 input cells.

A generalization of this result to all m-types (see Table S4) showed that PCs require about $\approx 10-15$ samples, while interneurons require $\approx 5-10$ cells to approximate the distributions of their dendritic morphometrics with synthesized cells. Note that this is a generalization as for each specific cell type the

Cell Reports *39*, 110586, April 5, 2022 9

Attention Context Window



Cell Reports
Article

CellPress
OPEN ACCESS

optimize this electrical model are shown as a reference (in black). Several electrical features are extracted from each trace and compared against a distribution of features from experimental recording using a *Z* score (Figure 4G). All the *Z* scores of a given cell obtained under various protocols provide a robust indication of the quality of the electrophysiological behavior of the cell (Markram et al., 2015; Van Geit et al., 2016).

To assess the overall quality of the electrical simulations, the MVS score, as defined for the morphological features (STAR Methods: Population-to-population validation) is applied for the comparison of the electrical feature distributions of reconstructed and synthesized cells (Figure S8B). The synthesized cells perform as well as the reconstructed cells in the electrical simulations. However, some m-types with only few example reconstructions result in several high-scoring features for a subset of e-types (see also comment on morphological validation). The statistical results of the electrical validations are presented in detail in Table S5 and Figure S6A. In addition, we cross-validate the electrical simulations by applying the same e-models on two distinct populations of neurons (reconstructed and synthesized). We demonstrate that the synthesized cells are sufficiently similar to reconstructed cells to reproduce the variance and differences for two e-models (L3 and L5 PC) applied on two distinct populations of L3 and L5 pyramidal cells (see Figure S6B).

Morphological feature correlations

Correlations between morphological features are reportedly essential for any synthesis method (López-Cruz et al., 2011). However, do the inputs to TMD-based synthesis suffice to account for feature correlations, or do we need to define them explicitly? In previous studies, the explicit description of correlated morphometrics was either obtained manually (Koene et al., 2009; van Pelt and van Ooyen, 2013), with the lurking danger that different neuronal datasets might require different feature correlations, or optimized with complex algorithms (López-Cruz et al., 2011), which entails the risk of over-fitting when only a few reconstructions are available. The risk is that, instead of capturing the biological principles of neuronal morphologies, the algorithm might overestimates local and noisy properties.

The inputs for the TNS are defined by the topological barcodes of neurons. Barcodes encode path distances in the neuronal tree, but values such as radial distances, asymmetry, and the precise connectivity of the tree are not encoded in the barcodes. To ensure that the synthesis algorithm reproduces inter-dependencies between features, it is necessary to validate the distributions of the features that have not been explicitly used as inputs for the synthesis algorithm. However, as seen in the thorough statistical validation of morphometrics, synthesized cells do not differ from the biological reconstructions for an extensive list of morphological features, most of which were not direct input to the algorithm.

A necessary condition for this is the coupling between bifurcation and termination probabilities that is encoded in the barcode structure. If bifurcation and termination probabilities in reconstructed cells were in fact independent, this algorithm would suffice to reproduce the branching patterns of the neuronal morphologies (Luczak, 2006; Cuntz et al., 2010). The TNS algorithm was modified to sample independently bifurcation and termination probabilities and bifurcation angles, instead of the original bars (see Figure S4). The TNS algorithm was modified to investigate the impact of using marginal branching probabilities instead of joint branching probabilities through persistence barcodes. To do this, we use the information encoded in the persistence barcode (b_i, d_i, a_i) as independent variables. Cells were synthesized by decoupled start and end of barcodes (Figure S4III), by decoupling start distance and branching angles (Figure S4IV). As shown in Figure S4, the synthesized cells generated by these versions of modified TNS algorithm are significantly different from the original reconstructions, indicating that the coupling between bifurcation and termination probabilities provides a necessary condition to reproduce correlations between morphological features.

We have thus demonstrated that the persistence barcodes encode the relevant information about the biological branching structures required for the accurate generation of dendritic shapes. In addition, the links between the bifurcation, the termination, and the respective bifurcation angles, are essential for synthesizing biologically accurate cells. The persistence barcodes are therefore a necessary constraint on the synthesis inputs.

Morphological diversity

Another challenge in synthesis is the sparsity of input data for many cell types, which makes it difficult to reproduce the morphological diversity of neurons. If few biological reconstructions are available (fewer than five cells), it is impossible to reproduce the properties of this cell type in its *in vivo* conditions. Using groups of cells with a large number of available reconstructions, such as PCs of layers 3–5, we investigated how the number of input cells influences the diversity of synthesized morphologies in Figure S5.

In particular, we compared the synthesized distributions of fundamental morphological features, such as path distance, branch order, and radial distance for increasingly larger subpopulations of 44 L4_TPC reconstructed cells. We identified the minimum number of cells that is required as input for synthesis to approximate well the morphological features of the original reconstructed population. While a sample size of ≤ 10 cells was not sufficient to approximate the diversity of the reconstructed cells with respect to our choice of morphometrics, with ≥ 15 (one-third of the original dataset), both input and emergent morphometrics were well reproduced (see Figures S9A, S9B, S9D, and S9E). Note that, since morphological features are intrinsically related to each other, we cannot define purely emergent properties, but rather properties that were not directly used as input to the algorithm. In addition, we computed the average KS distance between the distributions of reconstructed and synthesized cells for all features (Figure S9C) to confirm that the KS distance is minimized for both basal and apical features with more that 10−15 input cells.

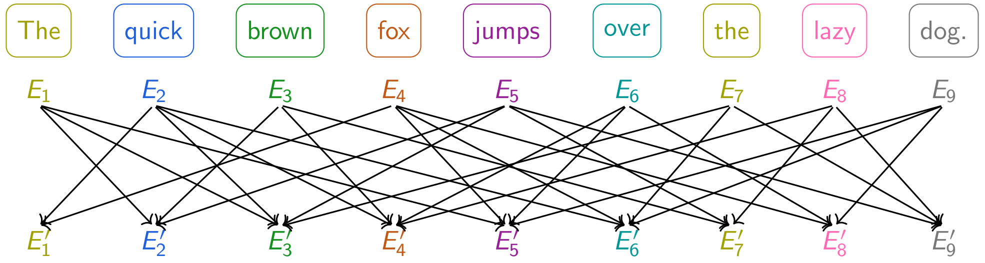
A generalization of this result to all m-types (see Table S4) showed that PCs require about $\approx 10-15$ samples, while interneurons require $\approx 5-10$ cells to approximate the distributions of their dendritic morphometrics with synthesized cells. Note that this is a generalization as for each specific cell type the

Cell Reports *39*, 110586, April 5, 2022 9

Attention transformation

$$\begin{tikzcd}
\boxed{\text{The}} & \boxed{\text{quick}} & \boxed{\text{brown}} & \boxed{\text{fox}} & \boxed{\text{jumps}} & \boxed{\text{over}} & \boxed{\text{the}} & \boxed{\text{lazy}} & \boxed{\text{dog.}} \\
E_1 \arrow[d] & E_2 \arrow[d] & E_3 \arrow[d] & E_4 \arrow[d] & E_5 \arrow[d] & E_6 \arrow[d] & E_7 \arrow[d] & E_8 \arrow[d] & E_9 \arrow[d] \\
E_1' & E_2' & E_3' & E_4' & E_5' & E_6' & E_7' & E_8' & E_9'
\end{tikzcd}$$

Attention transformation


The
quick
brown
fox
jumps
over
the
lazy
dog.
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9
E'_1 E'_2 E'_3 E'_4 E'_5 E'_6 E'_7 E'_8 E'_9

Attention Query

The same matrix W_Q is used to perform the query transformations on all embedding vectors E_i which are transformed into the respective Q_i

$$\begin{tikzcd}
\boxed{\text{The}} & \boxed{\text{quick}} & \boxed{\text{brown}} & \boxed{\text{fox}} & \boxed{\text{jumps}} & \boxed{\text{over}} & \boxed{\text{the}} & \boxed{\text{lazy}} & \boxed{\text{dog.}} \\
E_1 \arrow[d, "W_Q"] & E_2 \arrow[d, "W_Q"] & E_3 \arrow[d, "W_Q"] & E_4 \arrow[d, "W_Q"] & E_5 \arrow[d, "W_Q"] & E_6 \arrow[d, "W_Q"] & E_7 \arrow[d, "W_Q"] & E_8 \arrow[d, "W_Q"] & E_9 \arrow[d, "W_Q"] \\
Q_1 & Q_2 & Q_3 & Q_4 & Q_5 & Q_6 & Q_7 & Q_8 & Q_9
\end{tikzcd}$$

Attention Query (Q)

Let $E_t \in \mathbb{R}^d$ be the token representation at position t. We compute a **query** via a learned map:

$$Q_t = W_Q\, E_t, \qquad W_Q \in \mathbb{R}^{d_k \times d}.$$

Interpretation:

- Q_t encodes what information position t seeks from the context.
- d_k is the query/key dimension (commonly $d_k = d/\text{heads}$).

Attention Query (Q)

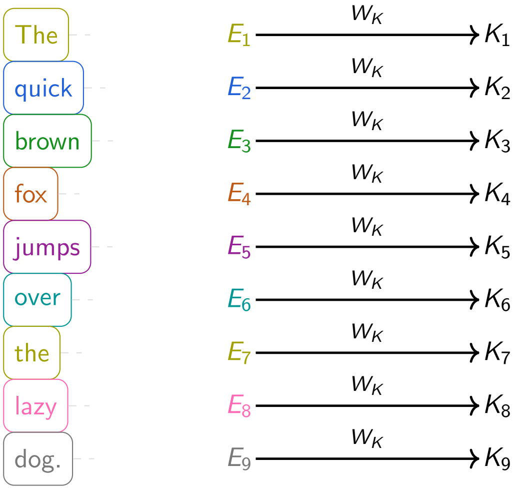
Example query: “Which adjectives describe the noun ’cat’ in this sentence?” When E_t corresponds to the token “cat”, Q_t will ask for tokens that are adjective-like and nearby. The attention mechanism uses this Q_t to score all keys and retrieve useful values.

Attention Key

The same matrix W_K is used to perform the key transformations on all embedding vectors E_i which are transformed into the respective K_i

$$\begin{tikzcd}
\boxed{\text{The}} & \boxed{\text{quick}} & \boxed{\text{brown}} & \boxed{\text{fox}} & \boxed{\text{jumps}} & \boxed{\text{over}} & \boxed{\text{the}} & \boxed{\text{lazy}} & \boxed{\text{dog.}} \\
E_1 \arrow[d, "W_K"] & E_2 \arrow[d, "W_K"] & E_3 \arrow[d, "W_K"] & E_4 \arrow[d, "W_K"] & E_5 \arrow[d, "W_K"] & E_6 \arrow[d, "W_K"] & E_7 \arrow[d, "W_K"] & E_8 \arrow[d, "W_K"] & E_9 \arrow[d, "W_K"] \\
K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_8 & K_9
\end{tikzcd}$$

Attention Key



$$E_1 \xrightarrow{W_K} K_1$$
$$E_2 \xrightarrow{W_K} K_2$$
$$E_3 \xrightarrow{W_K} K_3$$
$$E_4 \xrightarrow{W_K} K_4$$
$$E_5 \xrightarrow{W_K} K_5$$
$$E_6 \xrightarrow{W_K} K_6$$
$$E_7 \xrightarrow{W_K} K_7$$
$$E_8 \xrightarrow{W_K} K_8$$
$$E_9 \xrightarrow{W_K} K_9$$

Attention Key (K): How Well Items Match the Query

Key vectors:

$$K_i = W_K E_i, \qquad W_K \in \mathbb{R}^{d_k \times d}.$$

Role: Each key describes what information a token can provide. The similarity between a query Q_t and a key K_i measures how relevant token i is to token's request.

Matching Query to Keys

$$\begin{tikzcd}
Q_t \arrow[r, "Q^\top K_1"] \arrow[dr, "Q^\top K_2"] \arrow[ddr, "Q^\top K_3"'] & K_1 \\
& K_2 \\
& K_3
\end{tikzcd}$$

Compute compatibility scores $Q^\top K_i$ followed by scaling and softmax.

Attention Key (K)

If Q_t asks "Is this token an adjective describing the nearby noun?" then a matching key K_i from a token like "fluffy" will yield a high dot product $Q_t^\top K_i$, causing attention to pick up the value of that token.

Attention: Query and Key concepts

	The	quick	brown	fox	jumps	over	the	lazy
The								
quick								
brown								
fox								
jumps								
over								
the								
lazy								

Attention: Query and Key concepts

The quick brown fox jumps over the lazy

The

quick

brown

fox

jumps

over

the

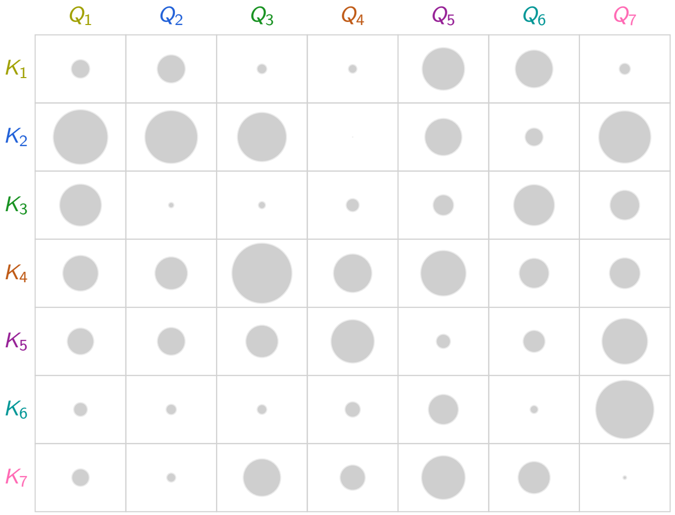
lazy

$$
\begin{array}{c|cccccccc}
 & K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_8 \\
\hline
Q_1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
Q_8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}
$$

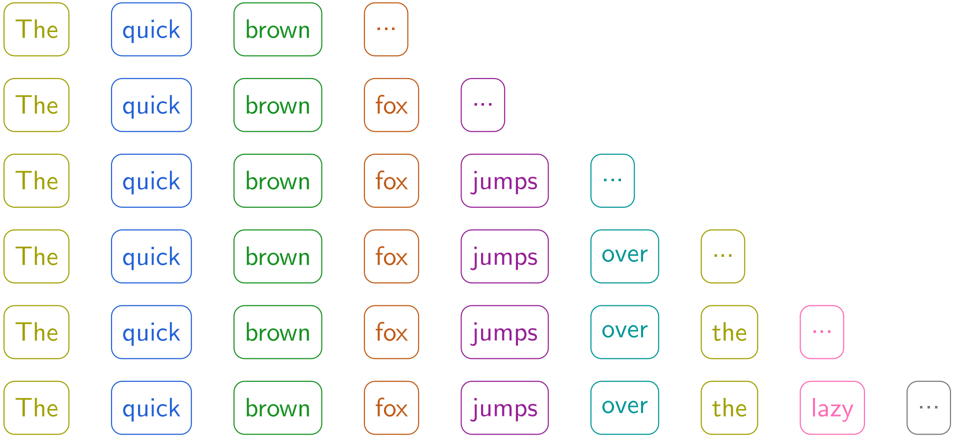
Attention: Query and Key concepts

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
K_1	$K_1 \cdot Q_1$	$K_1 \cdot Q_2$	$K_1 \cdot Q_3$	$K_1 \cdot Q_4$	$K_1 \cdot Q_5$	$K_1 \cdot Q_6$	$K_1 \cdot Q_7$
K_2	$K_2 \cdot Q_1$	$K_2 \cdot Q_2$	$K_2 \cdot Q_3$	$K_2 \cdot Q_4$	$K_2 \cdot Q_5$	$K_2 \cdot Q_6$	$K_2 \cdot Q_7$
K_3	$K_3 \cdot Q_1$	$K_3 \cdot Q_2$	$K_3 \cdot Q_3$	$K_3 \cdot Q_4$	$K_3 \cdot Q_5$	$K_3 \cdot Q_6$	$K_3 \cdot Q_7$
K_4	$K_4 \cdot Q_1$	$K_4 \cdot Q_2$	$K_4 \cdot Q_3$	$K_4 \cdot Q_4$	$K_4 \cdot Q_5$	$K_4 \cdot Q_6$	$K_4 \cdot Q_7$
K_5	$K_5 \cdot Q_1$	$K_5 \cdot Q_2$	$K_5 \cdot Q_3$	$K_5 \cdot Q_4$	$K_5 \cdot Q_5$	$K_5 \cdot Q_6$	$K_5 \cdot Q_7$
K_6	$K_6 \cdot Q_1$	$K_6 \cdot Q_2$	$K_6 \cdot Q_3$	$K_6 \cdot Q_4$	$K_6 \cdot Q_5$	$K_6 \cdot Q_6$	$K_6 \cdot Q_7$
K_7	$K_7 \cdot Q_1$	$K_7 \cdot Q_2$	$K_7 \cdot Q_3$	$K_7 \cdot Q_4$	$K_7 \cdot Q_5$	$K_7 \cdot Q_6$	$K_7 \cdot Q_7$

Attention: Query and Key concepts — circle heatmap



Attention transformation


The quick brown ⋯
The quick brown fox ⋯
The quick brown fox jumps ⋯
The quick brown fox jumps over ⋯
The quick brown fox jumps over the ⋯
The quick brown fox jumps over the lazy ⋯

Attention masking

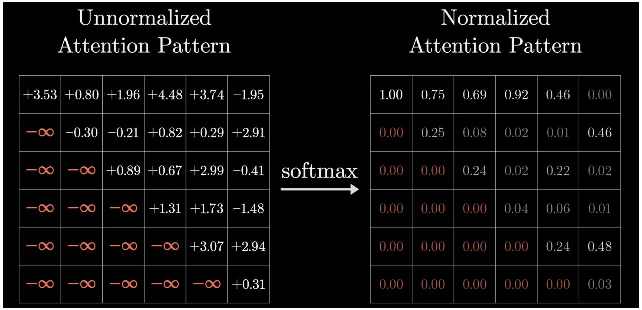
For autoregressive generation we must prevent peeking at future tokens. Implement mask:

$$\text{score} = \begin{cases} Q_t^\top K_i/\sqrt{d_k}, & i \le t, \\ -\infty, & i > t \end{cases}$$

After softmax this yields a lower-triangular attention matrix (causal mask).

Attention masking

Due to the iterative nature of the prediction algorithm, we want the model to only have access to the previous words. Therefore we need to only keep the upper triangular matrix. For correct normalization, we set the values to $-\infty$ before applying the softmax algorithm.



Unnormalized Attention Pattern

$+3.53$	$+0.80$	$+1.96$	$+4.48$	$+3.74$	-1.95
$-\infty$	-0.30	-0.21	$+0.82$	$+0.29$	$+2.91$
$-\infty$	$-\infty$	$+0.89$	$+0.67$	$+2.99$	-0.41
$-\infty$	$-\infty$	$-\infty$	$+1.31$	$+1.73$	-1.48
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$+3.07$	$+2.94$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$+0.31$

$\xrightarrow{\text{softmax}}$

Normalized Attention Pattern

1.00	0.75	0.69	0.92	0.46	0.00
0.00	0.25	0.08	0.02	0.01	0.46
0.00	0.00	0.24	0.02	0.22	0.02
0.00	0.00	0.00	0.04	0.06	0.01
0.00	0.00	0.00	0.00	0.24	0.48
0.00	0.00	0.00	0.00	0.00	0.03

Attention queries and keys

The attention matrix

$$K^\top Q$$

allows us to define which words are relevant to influence other words.

Attention queries and keys

The attention matrix

$$K^\top Q$$

allows us to define which words are relevant to influence other words.
But how can we update our embeddings accordingly to which words influence the context?

Attention value matrix

The value matrix

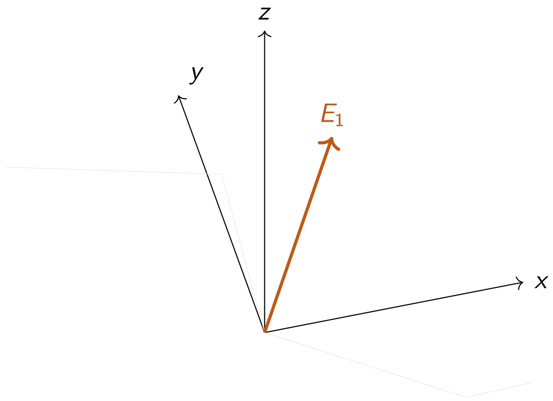
$$W_V$$

is multiplied by the vector of the context word and added to our initial embedding. For example the embedding E_4 of the fox is updated by adding the value of the adjectives that describe it $W_V E_3$ for which the $K_3 Q_3$ attention value is high.

$$E_4' \approx E_4 + \mathrm{softmax}(K_3 Q_3) V_3$$

Embedding update

How to update the initial embedding?



Attention value matrix

Adding all the values from the attention context we get the overall difference to be added to the embedding vector:

$$\Delta E_4' = \sum_i \mathrm{softmax}\left(\frac{K_i Q_i}{\sqrt{d_k}}\right) V_i$$

And the new embedding is updated to reflect the context

$$E_4' \approx E_4 + \Delta E_4$$

Attention Value (V): What is Returned

Values are computed as

$$V_i = W_V E_i, \qquad W_V \in \mathbb{R}^{d_v \times d}.$$

Role: V_i contains the information to aggregate (a content vector). The attention weights α_i select and mix these values into a context vector for each query.

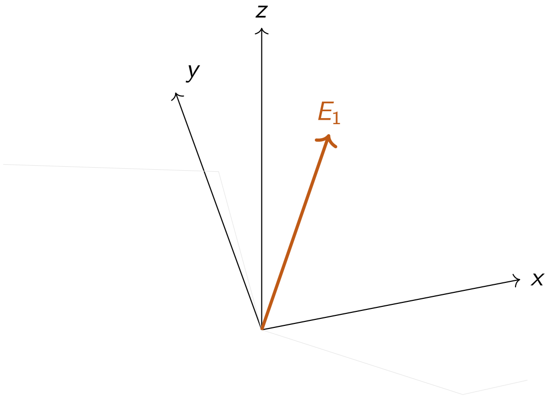
How Attention Updates the Embedding

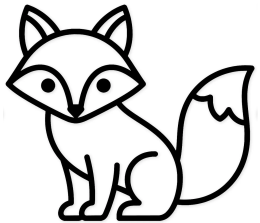
The attention output for position t is

$$a_t = \sum_{i=1}^{n} \alpha_{ti} v_i.$$

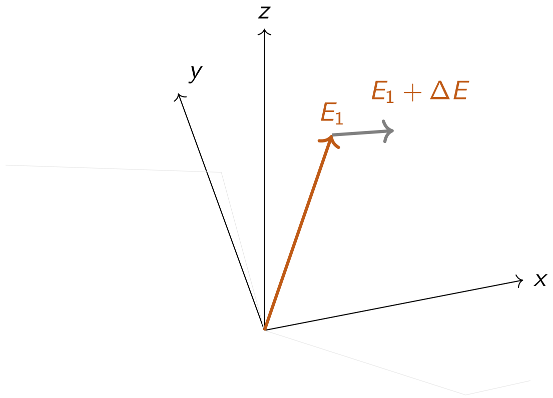
This a_t is combined (often via residual connection and layer norm) with the original E_t to produce an updated representation that encodes contextual information — effectively moving E_t in embedding space toward context-relevant directions.

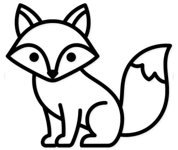
How to update the initial embedding?



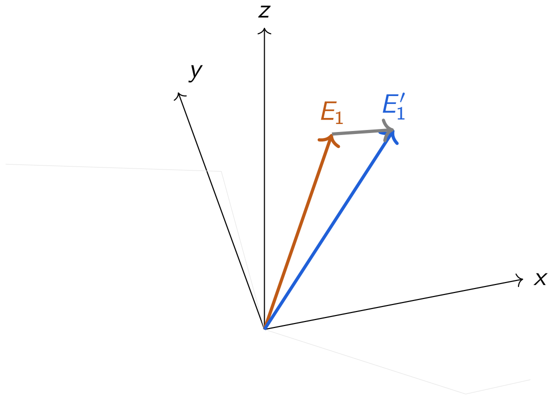


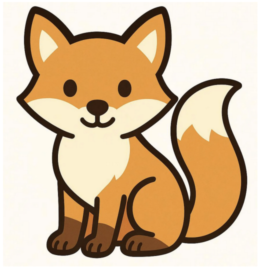
How to update the initial embedding?





How to update the initial embedding?





Attention Context Window

The attention matrix

$$K^\top Q$$

depends on the context size *cont* and it is in fact $\approx (cont)^2$. This is why it's important to choose context size appropriately.
For GPT-3:

$$\text{Context window} = 2048 \text{ tokens.}$$

This defines how much prior text the model can "see" when predicting the next word.

Computational Complexity

Attention computes an $n \times n$ matrix of pairwise scores (for n tokens) $\rightarrow$ required memory / compute grows as $\mathcal{O}(n^2)$.

Implication: large context windows (e.g. 2048 tokens) can be expensive; many research efforts focus on reducing this bottleneck.

Attention Overview

Next, we study how transformers use attention to focus on relevant words in context.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{K^\top Q}{\sqrt{d_k}}\right) V$$

Idea: each token attends to others based on similarity between queries and keys.

Attention

Given
queries $Q = [q_1, \ldots, q_n]$,
keys $K = [k_1, \ldots, k_n]$,
values $V = [v_1, \ldots, v_n]$:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{K^\top Q}{\sqrt{d_k}}\right) V.$$

Equivalently for single query q:

$$\mathrm{att}(q, K, V) = \sum_{i=1}^{n} \alpha_i v_i, \qquad \alpha = \mathrm{softmax}\left(\frac{K^\top q}{\sqrt{d_k}}\right).$$

Attention Single Head

What is attention?
A mechanism that lets each token dynamically re-weight (attend to) other tokens in the context according to relevance.

High-level idea:

- For each token we compute a *query* vector that asks “what am I looking for?”
- For each token we compute a *key* vector that answers “what do I have?”
- A compatibility score between query and key determines how much information (value) to read.
- The resulting weighted sum of values produces a context-aware representation.

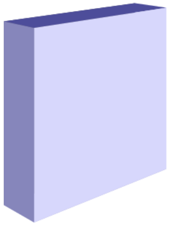
Self-Attention versus Cross-Attention

When the model conditions on another sequence (e.g., encoder–decoder, translation), we use cross-attention:

$$\mathrm{Attention}(Q_{\mathsf{dec}}, K_{\mathsf{enc}}, V_{\mathsf{enc}}).$$

The difference here, is that the key and query maps act on different datasets. There is typically no masking and the keys and queries map which elements of one dataset correspond to elements of the other dataset. For example, in translation this will correspond to matching between words in the two languages.
Use cases: translation, text-to-image, text-to-sound.

Single-Head Attention


Attention head

Overall, one attention layer transforms each embedding (of a token) to incorporate one aspect of the context. For example, give me all adjectives describpbing a noun.

Multi-Head Attention

Rather than a single attention, we compute H parallel attention heads:

$$\mathsf{head}_j = \mathrm{Attention}(QW_Q^{(j)}, KW_K^{(j)}, VW_V^{(j)}),$$

then concatenate heads and project:

$$\mathsf{MultiHead}(Q, K, V) = W_O[\mathsf{head}_1; \ldots; \mathsf{head}_H].$$

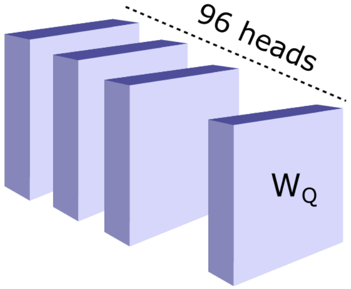
Benefit: different heads can attend to different relations / subspaces simultaneously.

Schematic: Multi-Head Attention

$$\begin{tikzcd}
\begin{matrix}\text{Input}\\ H \times d\end{matrix} \arrow[r] & \begin{matrix}\text{Split into}\\ \text{heads}\end{matrix} \arrow[r] & \begin{matrix}\text{Parallel}\\ \text{Attention Heads}\end{matrix} \arrow[r] & \begin{matrix}\text{Concatenate}\\ \text{and Project}\end{matrix}
\end{tikzcd}$$

Heads compute complementary attention patterns in different subspaces.

Multi-Head Attention



Multi-head attention

Putting multiple attention layers together transforms each embedding (of a token) to incorporate the full context.

Attention Overview

Attention consists of three main components:

- Query: Defines a question
- Key: Responses to the question
- Value: Updates the embedding by a value if the query is positive to the key.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{K^\top Q}{\sqrt{d_k}}\right) V$$

Each token attends to others based on similarity between queries and keys.
For each attention head the embedding is updated through a new query and at the end of the process, the new embedding reflects all the previous context.

Multi-Layer Perceptron

MLP in Transformer

For each vector (embedding), an MLP consists of two linear layers and one nonlinear ReLU:

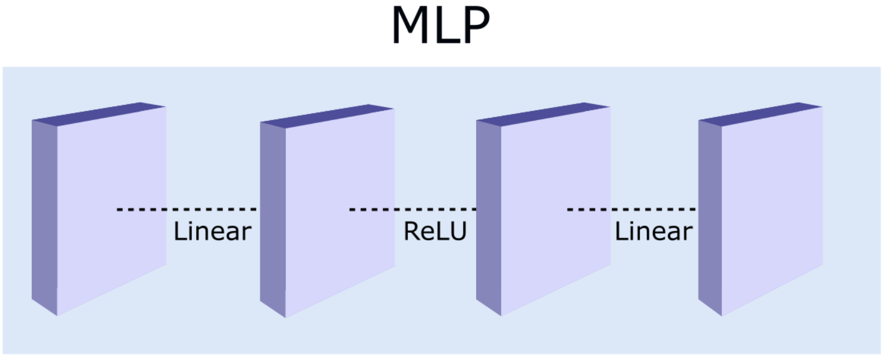
$$\mathrm{MLP}(x) = W_2\,\sigma(W_1 x + b_1) + b_2,$$

commonly $\sigma = \mathrm{ReLU}$.

MLP (Linear $\to$ ReLU $\to$ Linear)

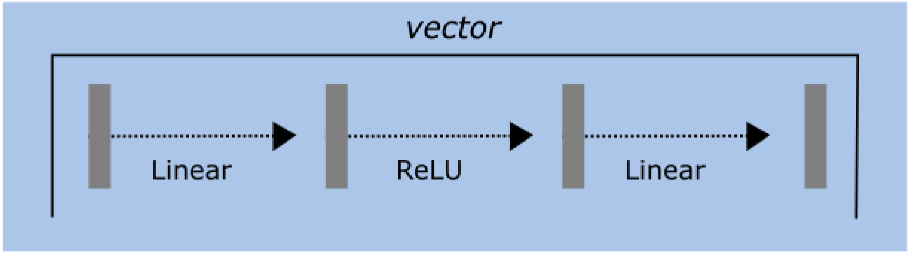
$$\begin{tikzcd}
\text{Input } x \in \mathbb{R}^d \arrow[r] & \begin{matrix} W_1 x + b_1 \\ \text{(linear)} \end{matrix} \arrow[r] & \begin{matrix} \sigma(x\prime) \\ \text{(ReLU)} \end{matrix} \arrow[r] & \begin{matrix} W_2 x\prime + b_2 \\ \text{(linear)} \end{matrix}
\end{tikzcd}$$

Multi-Layer Perceptron



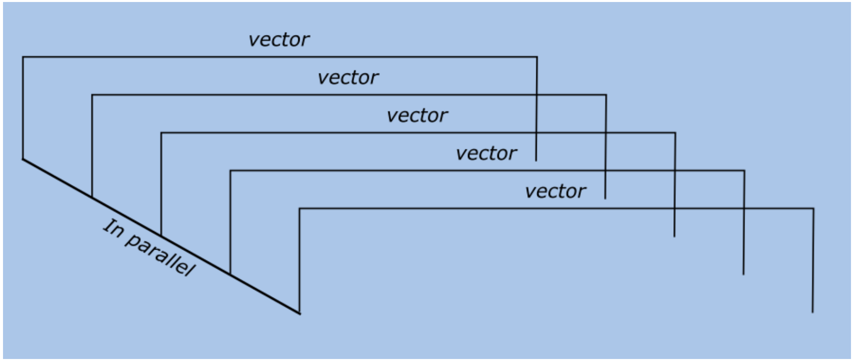
MLP computes alignment between each vector and a stored "memory" in the network.

Multi-Layer Perceptron



Each vector (embedding of a token) is processed independently.

Multi-Layer Perceptron



The vectors (embeddings tokens) are processed independently and in parallel through the MLP.

MultiLayer Perceptron principles

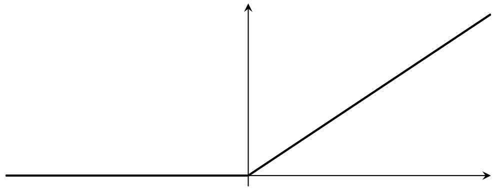
$$\begin{bmatrix}0.12\\-0.45\\0.30\\0.78\\-0.10\\0.05\\-0.22\end{bmatrix} \xrightarrow{\text{Linear}} \begin{bmatrix}-0.08\\0.60\\-0.09\\0.12\\0.44\\-0.33\\0.02\end{bmatrix} \xrightarrow{\text{ReLU}} \begin{bmatrix}0.00\\0.87\\0.50\\0.00\\0.33\\0.10\\0.01\end{bmatrix} \xrightarrow{\text{Linear}} \begin{bmatrix}0.22\\-0.11\\0.04\\0.66\\0.03\\-0.07\\0.18\end{bmatrix}$$

Schematic illustration of an MLP architecture: linear layer $\rightarrow$ ReLU nonlinearity $\rightarrow$ linear layer.

Rectified Linear Unit (ReLU)

$$\mathrm{ReLU}(z) = \max(0, z).$$

- Simple, computationally cheap nonlinearity.
- Introduces sparsity (negative pre-activations set to zero).



Rectified Linear Unit (ReLU)

- ReLU acts as a gate: only strong positive inputs pass through.
- In MLPs, different linear combinations are tested and ReLU selects which ones are active for the token.
- When active, a linear “feature detector” contributes to the next representation.

Final Layer

After MLP output $y = W_2\sigma(W_1x + b_1) + b_2$, the transformer typically:

$$x' = \mathrm{LayerNorm}(x + y).$$

Residual connections stabilize gradients and ease training of deep stacks.

Putting It All Together: Transformer Block Schematic

$$\begin{tikzcd}
\text{Input } E \arrow[r] & \begin{array}{c}\text{Multi-Head}\\ \text{Attention}\end{array} \arrow[r] & \text{Add \& Norm} \arrow[r] & \text{MLP} \arrow[r] & \begin{array}{c}\text{Add \& Norm}\\ \text{Output}\end{array}
\end{tikzcd}$$

Residual connections and normalization wrap both attention and MLP sub-layers.

Putting It All Together

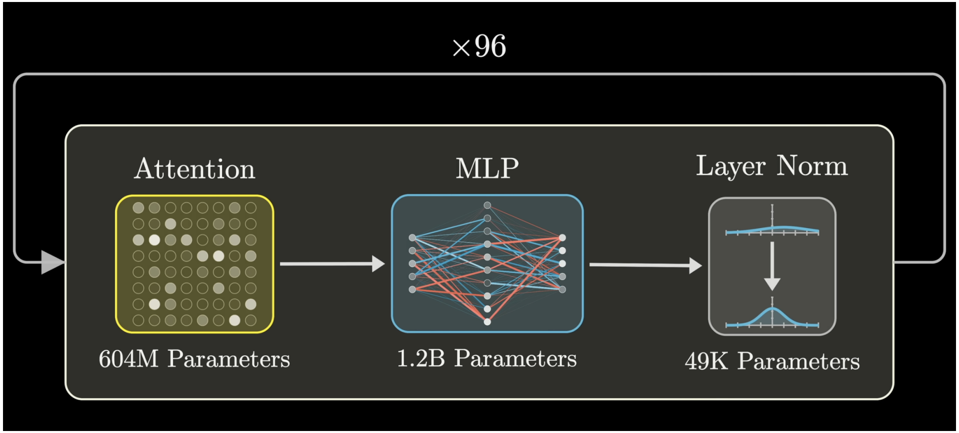


Figure: Example of LLM structure and counting parameters

Putting It All Together: From Tokens to Probabilities

1. Token $\rightarrow$ Embedding
2. Stack of Transformer blocks: each block updates E_t using attention $+$ MLP.
3. Final hidden E_T mapped to logits $z_T = W_U E_T$.
4. Softmax converts logits to probabilities $\mathrm{softmax}(z)$.

This pipeline is trained end-to-end to maximize next-token likelihood.

Estimate of parameters in an LLM

Process	Parameters
Embedding	$\approx 600M$
Query	$\approx 14B$
Key	$\approx 14B$
Value	$\approx 14B$
Output	$\approx 14B$
MLP (x96)	$\approx 116B$
Unembedding	$\approx 600M$
Total	$\approx 175B$

Summary

- Transformers use **attention** (Q, K, V) to compute context-aware representations.
- **Softmax** (temperature-scaled) converts logits to probabilities.
- **Multi-head attention** captures multiple relations in parallel.
- **MLP** acts like a gated memory function to identify per-position features (Linear–Nonlinearity–Linear).
- The full model maps tokens $\rightarrow$ embeddings $\rightarrow$ transformer layers $\rightarrow$ logits $\rightarrow$ probabilities and is trained via next-token prediction.

Thank you — Questions?