Introduction to Large Language Models

Large Language Models (LLMs) are advanced AI systems trained on massive text datasets to understand and generate human language with remarkable fluency. They power applications like chatbots, translation services, content summarisation, and code generation.

Modern LLMs, such as GPT-4, contain billions to trillions of parameters, enabling complex reasoning and sophisticated language tasks. The breakthrough Transformer architecture, introduced in 2017, revolutionised the field by using self-attention mechanisms for efficient, scalable learning.

Understanding LLM Architecture: The Transformer Model

Core Innovation

Transformers replaced older recurrent models with self-attention layers that process all tokens simultaneously, dramatically improving efficiency and enabling parallel computation at scale.

Embedding Layer

Converts words and tokens into high-dimensional numerical vectors that capture semantic meaning and relationships.

Multi-Head Self-Attention

Allows the model to focus on different parts of the input context simultaneously, capturing various linguistic patterns and dependencies.

Feed-Forward Networks

Add non-linearity and complexity through dense neural layers, enabling sophisticated feature transformations.

Stacked layers of these components build deep understanding and generation capabilities, with each layer refining representations further.

Query, Key, Value, and Output: The Attention Mechanism Explained

Query (Q)

Represents what the model is looking for. Derived from the input by multiplying with a learnable weight matrix. Asks: 'What information do I need?'

Key (K)

Represents the searchable content. Also derived from input with a different weight matrix. Answers: 'What information do I have?'

Value (V)

Contains the actual information to be retrieved. Transformed from input with its own weight matrix. Represents: 'The actual content to use'

Output Projection

Combines attended values into final representation. Another weight matrix transforms the concatenated attention heads into the output dimension. Produces: 'The refined context'

The attention mechanism computes similarity between Q and K to determine which values are most relevant, then outputs a weighted combination.

Scalars, Vectors, and Tensors: The Building Blocks

01

Scalar

A single numerical value representing a simple quantity (e.g., 5 or 3.14). Scalars are zero-dimensional.

02

Vector

A one-dimensional array of numbers representing features or word embeddings (e.g., [0.2, -0.5, 1.0]). Each dimension captures a specific attribute.

03

Tensor

A multi-dimensional array generalising scalars and vectors. A matrix is a 2D tensor; LLM inputs are typically 3D tensors structured as: batch size × sequence length × embedding dimension.

LLMs manipulate these tensors through matrix multiplications and transformations, combining information across dimensions to understand context and generate coherent text. Each mathematical operation preserves and enhances the semantic relationships encoded in the data.

Key Insight: The power of LLMs lies in their ability to efficiently process and transform high-dimensional tensors representing language.

What Are Parameters in LLMs?

Learnt Weights

Parameters are the model's learnt weights and biases that define how input data transforms through the network layers during inference.

Stored as Matrices

These parameters are stored in matrices (tensors) and adjusted during training to minimise prediction errors through backpropagation.

Model Capacity

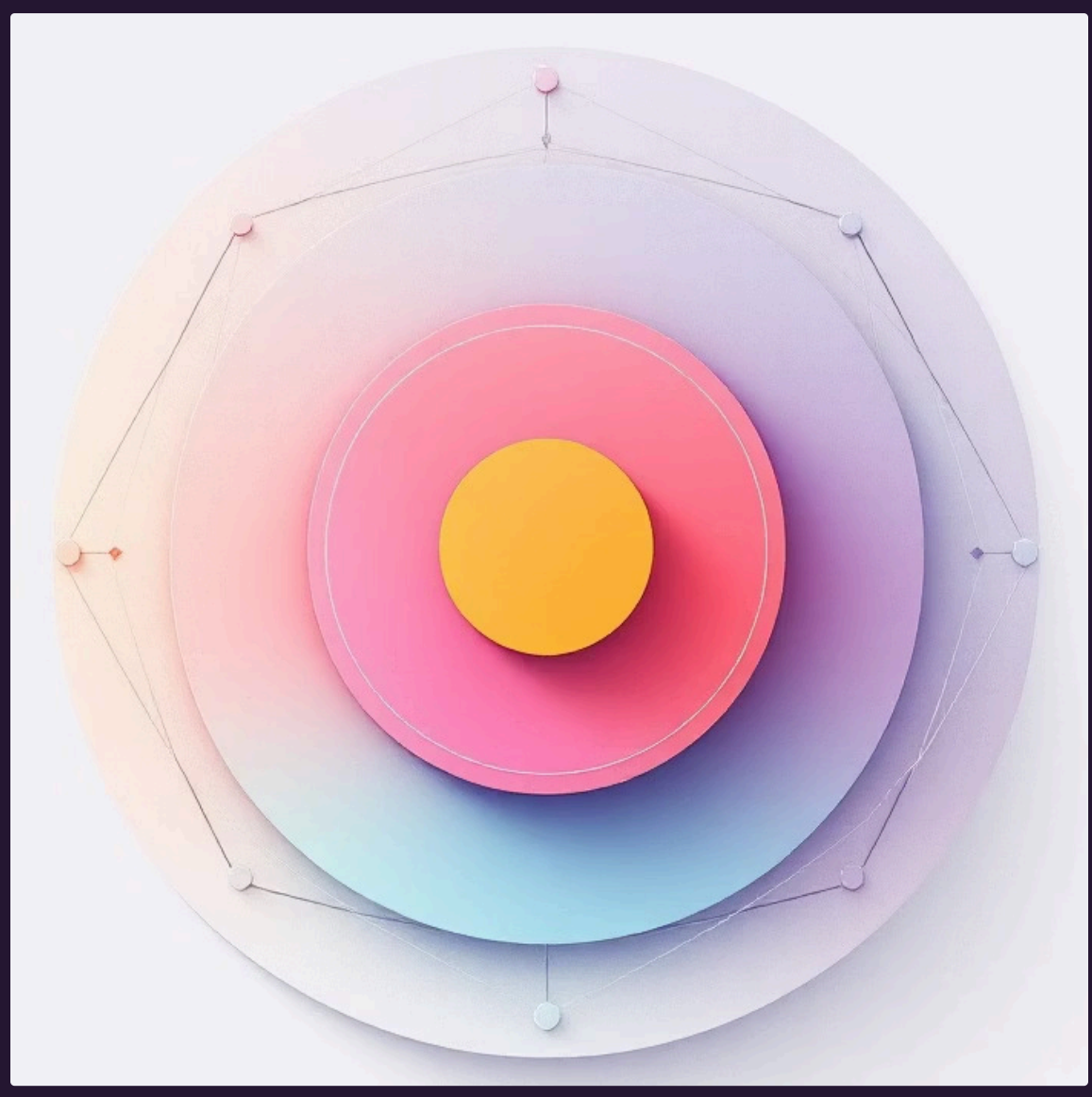
The number of parameters correlates directly with model capacity and complexity, determining what patterns the model can learn.

> Example: An embedding matrix size equals vocabulary size × embedding dimension. For a vocabulary of 50,000 words with 768-dimensional embeddings, that's 38.4 million parameters just for embeddings.

Computing Parameters: A Practical Example

Toy Model Specifications

- Vocabulary size: 10 words
- Embedding dimension: 10
- Transformer layers: 2
- Attention heads per layer: 2
- Feed-forward dimension: 12



Embedding Layer

10 × 10 = **100 parameters**

Attention Matrices

Each attention matrix (Q, K, V, Output): 10 × 10 = 100 parameters

4 × 100 = **400 parameters per layer**

Feed-Forward Network

2 × 10 × 12 + 10 + 12 = **262 parameters per layer**

Total Per Transformer Layer

400 + 262 = **662 parameters**

1,324

Two Transformer Layers

2 × 662 parameters

1,424

Total Model Parameters

100 (embedding) + 1,324

This arithmetic scales up to billions of parameters in real-world LLMs such as GPT-4, but follows exactly the same fundamental principles. Understanding this calculation helps demystify how these powerful models achieve their capabilities through sheer scale and sophisticated architecture.