How a Transformer Neural Network Works

A one-page primer on the architecture behind modern AI

What is a Transformer?

A transformer is a neural network that processes sequences — like words in a sentence — by letting every element *attend to* every other element in parallel. Introduced in *Attention Is All You Need* (Vaswani et al., 2017), it replaced slower recurrent models and became the backbone of LLMs such as GPT and BERT.

Key Components

- **Token Embedding:** each word/sub-word is mapped to a dense vector.
- **Positional Encoding:** adds order information, since attention is permutation-invariant.
- **Self-Attention:** computes how much each token should focus on every other token.
- **Feed-Forward Network (FFN):** a position-wise MLP that refines each token's representation.
- **Add & Norm:** a residual connection plus layer normalization, stabilizing training.

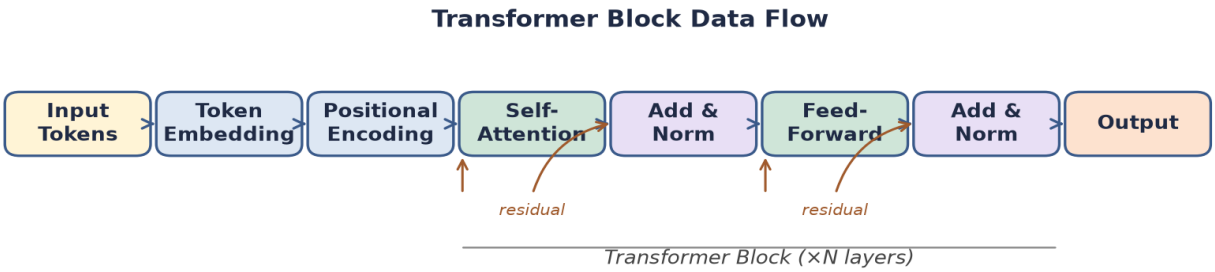
Self-Attention in One Equation

Each input vector *x* is projected into three: **Query (Q)**, **Key (K)**, and **Value (V)**. Attention is then a weighted sum:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T / \sqrt{d_k} \right) \cdot V$$

Dividing by $\sqrt{d_k}$ keeps gradients stable; softmax turns scores into a probability distribution over tokens. **Multi-head attention** runs several such computations in parallel, capturing different relationships.

Architecture at a Glance



Each block repeats N times. Residual arrows let gradients flow and preserve the original signal through the deep stack.

Stacking & Output

An **encoder** stack builds contextual representations of the input; a **decoder** stack generates outputs token-by-token, also using *cross-attention* to look back at the encoder. Pure-encoder models (BERT) understand text; pure-decoder models (GPT) generate it. Training uses massive corpora and next-token prediction, scaling to billions of parameters.