How a Transformer Neural Network Works

A concise guide to the architecture behind modern language models

Overview

A transformer is a neural network that processes sequences of tokens in parallel rather than one step at a time. It uses an **attention** mechanism to model relationships between all tokens simultaneously, making it both fast to train on parallel hardware and powerful at capturing long-range context.

Tokens & Embeddings

Text is split into tokens, each mapped to a learned vector called an **embedding**. Because attention alone has no sense of order, a **positional encoding** is added to every embedding so the model knows where each token sits in the sequence.

Self-Attention

Each token emits three learned vectors — a **query** (Q), a **key** (K), and a **value** (V). A token's query is compared (dot product) to every key to produce attention scores; after a softmax these become weights. The output for each token is a weighted sum of all values, letting it pull information from the most relevant tokens anywhere in the sequence.

Multi-Head Attention

Several attention heads run in parallel, each with its own Q/K/V projections. Different heads learn different relationships (syntax, coreference, topic). Their outputs are concatenated and projected back to a single vector.

Feed-Forward, Residuals & Norm

After attention, each position passes independently through a two-layer **feed-forward network**. **Residual** (skip) connections and **LayerNorm** wrap both sub-layers, stabilizing training and letting gradients flow through deep stacks.

Stacking & Output

This block is repeated N times (often 6–96). An encoder-only stack maps inputs to rich contextual vectors; a decoder adds masked attention for autoregressive generation. A final linear layer plus softmax turns the last hidden states into probabilities over the vocabulary, producing the next token.