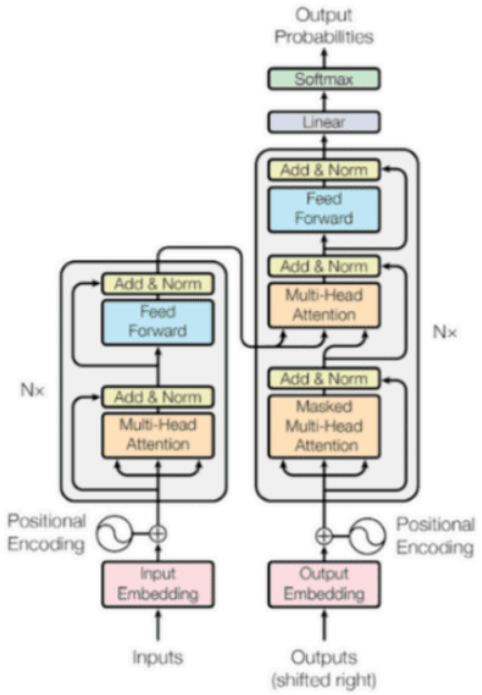
Transformers

Transformer

Attention Is All You Need



The Transformer is a revolutionary neural network architecture introduced in the 2017 paper "Attention Is All You Need." It has become the foundational model for modern large language models (LLMs) like GPT, BERT, and Gemini, and is highly effective across various domains, especially Natural Language Processing (NLP) but also in areas like computer vision (Vision Transformers) and audio processing.

Its primary innovation is relying entirely on a mechanism called **Self-Attention**, allowing it to process entire sequences in parallel, unlike its predecessors (like Recurrent Neural Networks/RNNs and Long Short-Term Memory/LSTMs) which had to process data sequentially.

Overall Architecture

The original Transformer model consists of two main parts, each typically stacked with N = 6 identical layers:

- **Encoder Stack**: Processes the input sequence (e.g., a sentence in English for translation). Its job is to map the input text into a sequence of continuous representations that capture the meaning and context.
- **Decoder Stack**: Processes the encoder's output and its own output (shifted right) to generate the final output sequence (e.g., the translated sentence in French).

Input & Output Processing

1. **Input Embeddings:** The first step is to convert the input tokens (words or sub-words) into dense numerical vectors called **embeddings**. These vectors capture the semantic meaning of the tokens.
2. **Positional Encoding (PE):** Since the self-attention mechanism processes all tokens simultaneously, it loses the information about the **order** of the words in the sequence. Positional Encoding is a vector that is added to each input embedding to inject this crucial sequential information, allowing the model to distinguish between "dog bites man" and "man bites dog."

The Attention Mechanism

Attention is a mechanism that allows the model to weigh the importance of different tokens in the input sequence when processing a specific token.
The fundamental operation of attention uses three learned vectors for each token:

- **Query:** Represents the token asking for information.
- **Key:** Represents the content/information that a token has to offer.
- **Value:** Contains the actual information to be extracted and passed along.

The **Attention Mechanism** is a cool trick that **Artificial Intelligence (AI)** models use to read and understand information, just like you do. Instead of trying to pay equal attention to *every single word* in a sentence, the AI learns to **focus on the most important parts** to figure out what's going on.

The process for a **single attention** “head” is:

- First the similarity between the **Query vector** of the current token and the **Key vectors** of all tokens is computed using a **dot product**.
- Then the scores are **scaled to stabilize** gradients during training.
- A **softmax function** is applied to **normalize** the scores into attention weights.
- The value vectors of all tokens are multiplied by their corresponding attention weights and then summed up. The **weighted sum** is the new context-aware vector representation for the query token.

Multi-Head Attention: Instead of one attention computation, the Transformer performs this process multiple times in parallel with different, learned Q, K, and V weight matrices. This allows the model to attend to different aspects of the input simultaneously (e.g., one head might focus on grammatical relationships, another on semantic meaning). The outputs of all heads are then concatenated and linearly transformed.

Layer Structure (Inside Encoder/Decoder Blocks)

Both the Encoder and Decoder layers contain:

1. **Multi-Head Attention:** (Self-Attention in the Encoder; Masked Self-Attention and Cross-Attention in the Decoder).
    a. **Encoder**: This layer allows the encoder to weigh the importance of all other words in the input sequence for each word being processed. This is how the model captures long-range dependencies and context.
    b. **Decoder**: Similar to the encoder's self-attention, but with one critical difference: it uses a **mask** to prevent each position from attending to subsequent positions (i.e., tokens that come *after* the current token in the output sequence). This is crucial for **auto-regressive generation**, ensuring that the prediction for a token depends only on the known tokens that came before it. The cross attention layer is where the decoder integrates the contextualized information from the encoder's output. It takes its **Queries (Q)** from the preceding masked self-attention sub-layer's output and its **Keys (K)** and **Values (V)** from the top encoder layer's output. This allows the decoder to selectively focus on relevant parts of the *input* sequence.

2. **Feed-Forward Network (FFN):** A simple, position-wise two-layer fully connected network applied independently and identically to each position in the sequence. It processes the information extracted by the attention layer.
    a. **Encoder**: This is a simple fully-connected feed-forward network applied to each position in the sequence independently and identically. It consists of two linear transformations with a ReLU activation in between. Its purpose is to process the contextual information gathered by the attention layer and transform the representation into a higher-level feature space.
    b. **Decoder**: Identical to the FFN in the encoder block, this layer processes the output of the cross-attention sub-layer.

Encoder: The Encoder's job is to read an input sentence and deeply understand the meaning and context of every word.

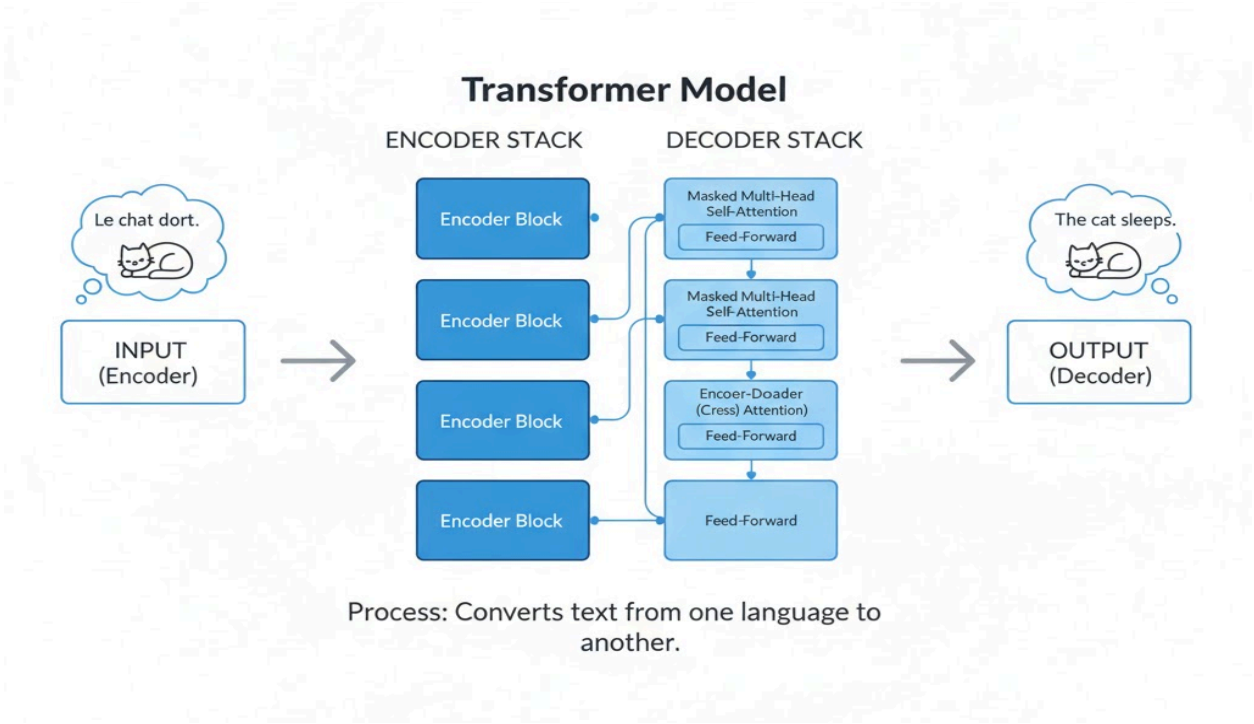
Decoder: The Decoder's job is to use the Encoder's deep understanding (the source language) and the words it has already generated (the target language) to predict the next word in the output sequence.

Advantages of Transformers

- **Parallelization:** The non-sequential nature (no RNNs) allows all tokens to be processed at the same time, significantly reducing training time and enabling the use of powerful GPUs.

- **Long-Range Dependencies:** Self-attention can directly connect any two tokens in a sequence, regardless of their distance, effectively capturing long-range dependencies that were challenging for RNNs/LSTMs.
- **Contextual Understanding:** The attention mechanism provides rich, context-aware vector representations for each token, leading to superior performance in complex NLP tasks like machine translation, text generation, and question answering.

Example



Input (Encoder): A sentence in the source language (e.g., French: "Le chat dort.").

Process (Encoder & Decoder): The **Encoder** reads and fully understands the French sentence. The **Decoder** then uses that understanding, along with the English words it has already generated, to predict the next word in the target language.

Output (Decoder): A translated sentence in the target language (e.g., English: "The cat sleeps.").