**Transformers - Overall Architecture**

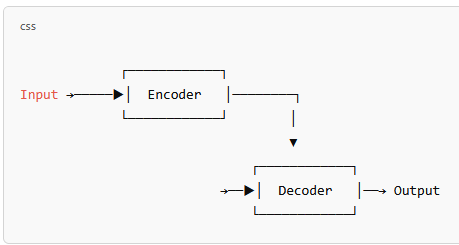
======================================================================

A **Transformer** is a deep learning model introduced in the paper *"Attention is All You Need"* (Vaswani et al., 2017).  
It relies entirely on **attention mechanisms** — no recurrence, no convolution — to model sequences.

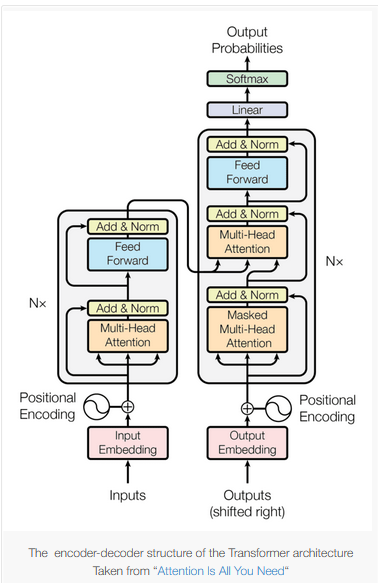
--------------------------------------------------------------------------------------------------------------

**High-Level Structure**

The Transformer is made of two main parts:



* **Encoder**: Reads the input (e.g., a sentence in English)
* **Decoder**: Generates the output (e.g., translated sentence in French)



--------------------------------------------------------------------------------------------------------------

**Inside Encoder:**

--------------------------------------------------------------------------------------------------------------

**1. Input Embedding**

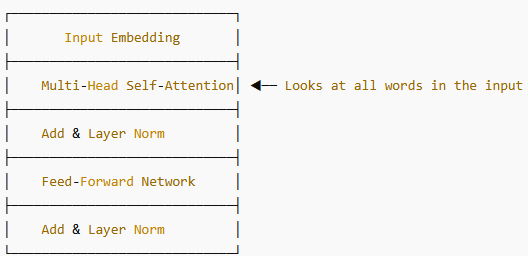
* Input tokens are converted to dense vectors (embeddings).

**2. Positional Encoding**

* Since Transformers don’t understand order, **positional encodings** are added.

**3. Encoder Block** (× N layers — usually 6 or 12)

Each encoder layer has:



All encoders **share the same structure** but not weights.

The encoder is responsible for processing the input sequence and creating a context-aware representation.

Each encoder layer has two main sub-layers:

* **Multi-Head Self-Attention:** This is the core of the Transformer. It allows each token in the input sequence to attend to all other tokens in the same sequence, calculating a weighted average based on their relevance to the current token. This enables the model to understand the context of each word by considering its relationships with other words in the sentence, regardless of their distance.

The "multi-head" aspect means that this self-attention mechanism is run multiple times in parallel with different learned linear projections, allowing the model to capture different types of relationships.

* **Position-wise Feed-Forward Networks:** After the self-attention layer, each token's representation is passed through a feed-forward network. This network consists of two linear transformations with a non-linear activation (usually ReLU) in between. It processes each position independently.
* **Residual Connections and Layer Normalization:** Around each of the two sub-layers (self-attention and feed-forward network), there is a residual connection followed by layer normalization.

Residual connections help with the flow of gradients in deep networks, and layer normalization stabilizes the learning process.

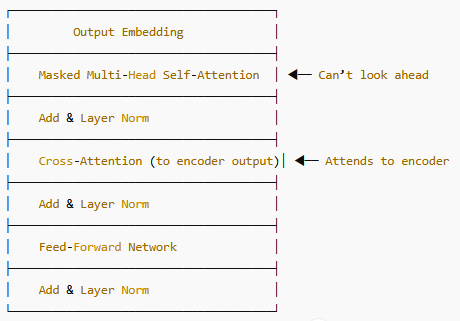
-------------------------------------------------------------------------------------------------------------

**Inside Decoder:**

--------------------------------------------------------------------------------------------------------------

**Decoder Block (× N layers — mirrors encoder)**

Each decoder layer has:



The decoder also uses **positional encoding** and shares the same attention structure.

The decoder is responsible for generating the output sequence based on the encoded representation from the encoder. It also consists of a stack of identical decoder layers. Each decoder layer has three main sub-layers:

* **Masked Multi-Head Self-Attention**: This is similar to the self-attention in the encoder, but with a crucial difference: it prevents the decoder from attending to future tokens in the output sequence during training.

This masking ensures that the prediction for each position only depends on the tokens generated before it, maintaining the autoregressive property of sequence generation.

* **Multi-Head Cross-Attention:** This layer receives the output of the encoder and the output of the previous decoder layer as inputs. It allows the decoder to attend to the relevant parts of the input sequence (encoded by the encoder) when generating the next token in the output sequence. The queries come from the decoder's previous layer, and the keys and values come from the encoder's output.
* **Position-wise Feed-Forward Networks:** Similar to the encoder, after the cross-attention layer, each token's representation is passed through a position-wise feed-forward network.
* **Residual Connections and Layer Normalization:** Again, residual connections and layer normalization are applied around each of these sub-layers.

--------------------------------------------------------------------------------------------------------------

**Final Linear Layer and Softmax:**

* The output of the final decoder layer is passed through a linear layer that projects the representation to the vocabulary size.
* A softmax function is then applied to produce a probability distribution over all possible output tokens. The token with the highest probability is chosen as the next token in the generated sequence.
* In training, we use teacher forcing (feeding the correct previous token).
* In inference, it generates token-by-token (auto-regressive).

--------------------------------------------------------------------------------------------------------------

**Key Concepts:**

* **Attention Mechanism:** The core idea that allows the model to focus on the most relevant parts of the input when processing or generating output.
* **Self-Attention:** Attention mechanism where a sequence attends to itself to understand internal relationships.
* **Multi-Head Attention**: Running the attention mechanism multiple times in parallel to capture different aspects of the relationships in the data.
* **Masked Attention**: Preventing attention to certain positions (typically future positions in the decoder) to maintain autoregressive generation.
* **Cross-Attention:** Attention mechanism where the decoder attends to the output of the encoder.
* **Residual Connections:** Skipping layers to help with gradient flow and training deeper networks.
* **Layer Normalization**: Normalizing activations within each layer to stabilize training.
* **Parallel Processing:** Unlike RNNs that process sequences sequentially, Transformers can process the entire input sequence in parallel, leading to faster training times.
* **Long-Range Dependencies**: Attention mechanisms allow Transformers to capture dependencies between tokens that are far apart in the sequence, overcoming a limitation of RNNs.

--------------------------------------------------------------------------------------------------------------

The Transformer architecture has become the foundation for many state-of-the-art models in NLP and has also found applications in other domains like computer vision (Vision Transformers - ViT) and audio processing. Its ability to model long-range dependencies and process data in parallel has made it a highly effective and efficient architecture for sequence-related tasks.