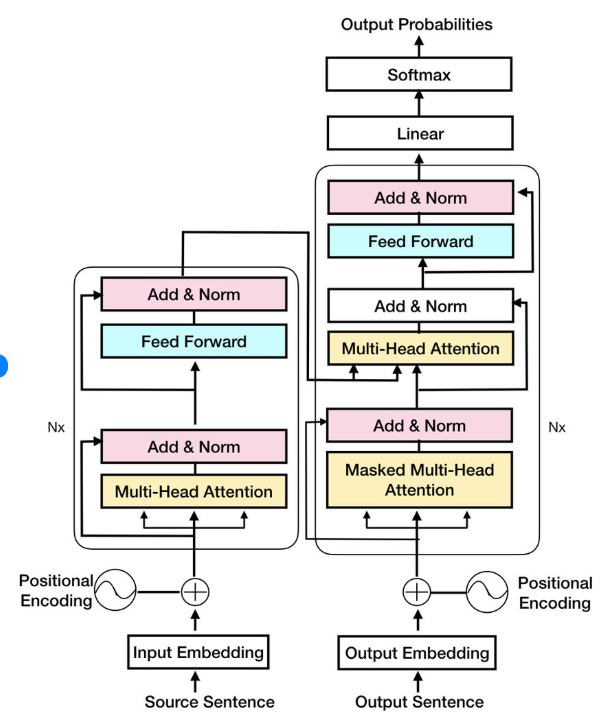
**Transformer Encoder Decoder**

The provided diagram illustrates the architecture of the **Transformer model**, a ground breaking neural network design introduced in the 2017 paper "Attention Is All You Need." It was initially designed for **sequence-to-sequence tasks** like machine translation, but its core mechanism, **self-attention**, has become the foundation for most modern large language models (LLMs) like BERT and GPT.

The architecture consists of two main parts: an **Encoder** (on the left) and a **Decoder** (on the right), each composed of a stack of identical layers, denoted by **Nx​** (meaning the block is repeated N times).

**The Encoder**

The **Encoder** processes the **Source Sentence** (e.g., the sentence in the language you want to translate *from*). Its goal is to create a rich, context-aware representation of the input.

**1. Input Processing**

* **Source Sentence:** The raw text input.
* **Input Embedding:** The source sentence is first converted into a sequence of **vectors** (numerical representations). Each word or token is mapped to a high-dimensional vector that captures its semantic meaning.
* **Positional Encoding:** Since the Transformer has no inherent recurrent (sequential) or convolutional components, it needs a way to understand the **order** of the words. **Positional Encodings** are added to the input embeddings. These are vectors that represent the position of each token in the sequence, allowing the model to know if a word is the first, second, or last, etc.
* **Positional Encoding**: Because Transformers don’t have recurrence, we add a **sinusoidal positional encoding** to each embedding to give the model a sense of word order.

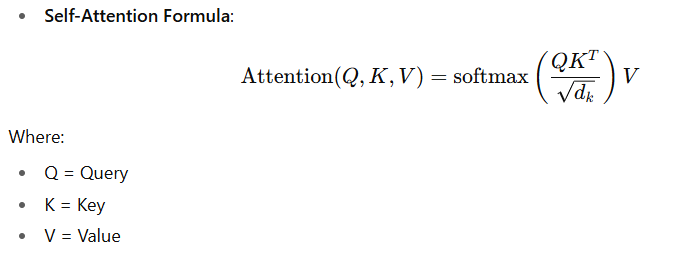
Input=Embedding+Positional Encoding

**2. Encoder Block (Repeated Nx​ times)**

Each encoder layer has two main sub-layers:

**A. Multi-Head Attention (Self-Attention)**

* The core of the Transformer. This mechanism allows every word in the input sequence to **attend to** (look at and weigh the importance of) all other words in the same sequence.
* It computes the relationships between words, which is crucial for understanding context (e.g., in the sentence "The animal didn't cross the street because **it** was too wide," the attention mechanism helps link "**it**" to "**street**" and not "**animal**").
* The "Multi-Head" part means the attention process is split into several independent "heads." Each head learns to focus on different aspects of the input, and their results are concatenated and linearly transformed.



**B. Feed Forward Network (FFN)**

* A simple, fully connected neural network (two linear transformations with a ReLU activation in between) applied **independently and identically** to each position (word vector). It allows the model to perform non-linear transformations on the attended information.

**3. Logic: Add & Norm (Residual Connections and Layer Normalization)**

* The diagram shows "Add & Norm" surrounding both the Multi-Head Attention and the Feed Forward Network.
* **"Add"** refers to a **Residual Connection** (or skip connection), where the input to a sub-layer is added to its output: Output=Input+Sublayer(Input). This helps prevent the gradients from vanishing during training, allowing for deep networks.
* **"Norm"** refers to **Layer Normalization**, which normalizes the activations across the features for all samples in a batch. This stabilizes the hidden state activations, accelerating training and improving performance.

**The Decoder**

The **Decoder** processes the **Output Sentence** (e.g., the partial or predicted sentence in the target language) and uses the representations from the Encoder to generate the final output, typically one token at a time.

**1. Input Processing**

* **Output Sentence:** The previously generated tokens (shifted right). During training, this is the correct target sentence shifted right (meaning the model only sees the tokens that came before the one it's predicting).
* **Output Embedding & Positional Encoding:** Similar to the Encoder, the partial output sentence is embedded, and Positional Encodings are added.

**2. Decoder Block (Repeated Nx​ times)**

Each decoder layer has three main sub-layers:

**A. Masked Multi-Head Attention (Self-Attention)**

* This is the same as the Encoder's self-attention, but with a critical difference: it is **Masked**.
* The masking operation prevents the model from looking at **future tokens** in the target sequence. This is essential for training and generation because at any given step, the model should only be able to attend to the words it has *already* generated (or the start of sequence token) to predict the next word.

**B. Multi-Head Attention (Encoder-Decoder Attention)**

* This is the key sub-layer that **connects the Encoder and the Decoder**.
* It performs attention over the **output of the final Encoder layer**.
* The Decoder's masked self-attention output acts as the **Query**, while the Encoder's output provides the **Keys and Values**.
* This mechanism allows the Decoder to focus on the **most relevant parts of the Source Sentence** to accurately generate the current token in the Output Sentence.

**C. Feed Forward Network (FFN)**

* Identical in function to the FFN in the Encoder, providing non-linear transformation.

**3. Logic: Add & Norm**

* As in the Encoder, Residual Connections and Layer Normalization are applied around all three sub-layers.

**Output Generation**

The output of the final Decoder layer is passed through the final transformation steps to produce a probability distribution over the entire vocabulary (all possible words/tokens).

* **Linear:** A simple **Linear transformation** (fully connected layer) maps the decoder output vector into a much larger vector, where the size equals the size of the model's vocabulary. Each element in this large vector represents the score (or *logit*) for a unique word in the vocabulary.
* **Softmax:** The **Softmax function** takes the logits and converts them into **Output Probabilities**—a probability distribution where all values are between 0 and 1 and sum up to 1. The token with the highest probability is then selected as the model's prediction for the next word in the sequence.



The entire process then repeats, using the newly predicted word as part of the input for the next step, until an end-of-sentence token is generated or a maximum length is reached.