Architecture and Working of Transformers in Deep Learning

Transformers are a type of deep learning model that utilizes **self-attention mechanisms** to process and generate sequences of data efficiently. They capture long-range dependencies and contextual relationships making them highly effective for tasks like language modeling, machine translation and text generation.

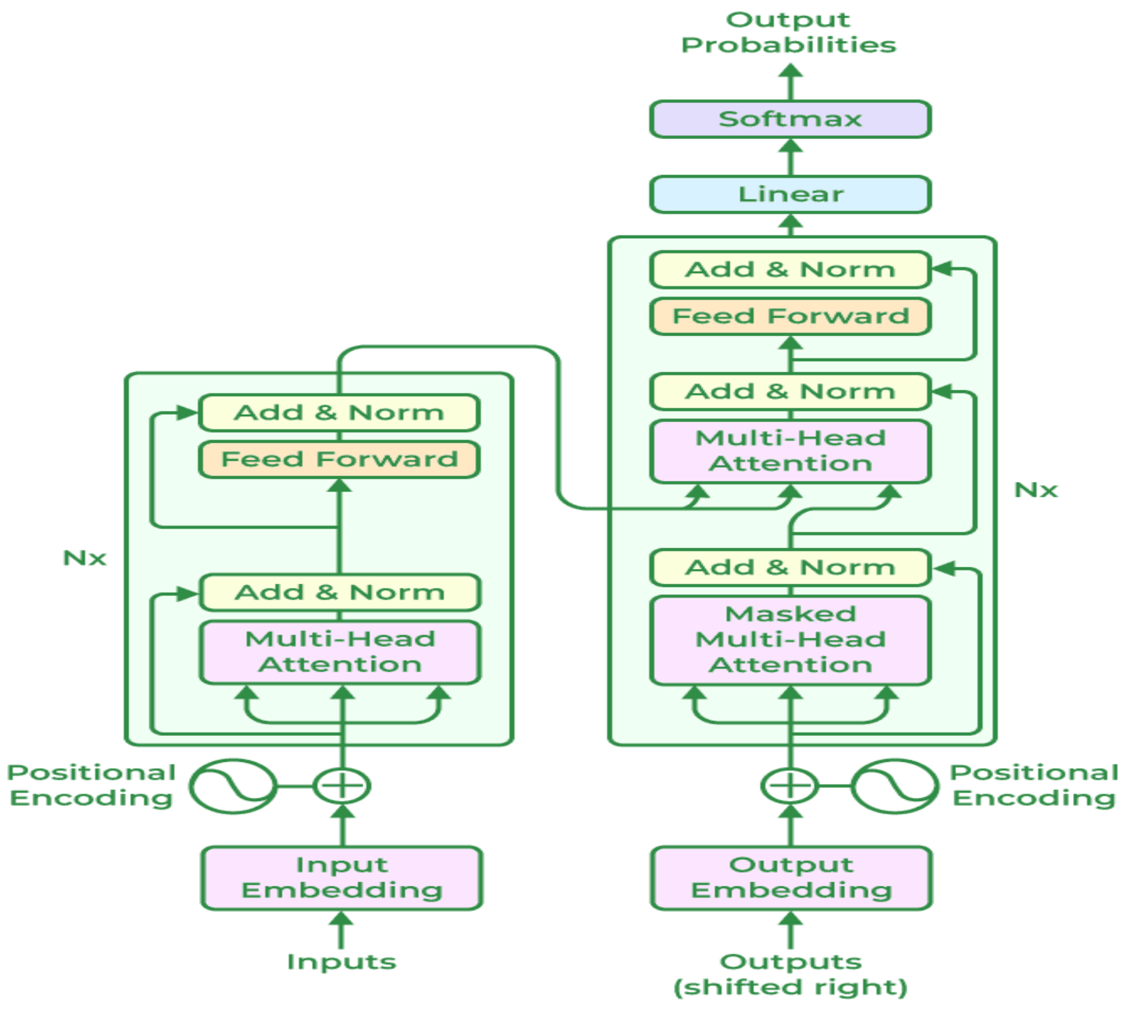
In this article we will explore the architecture and working of transformers by understanding their key components, mathematical formulations and how they function during training and inference.

Understanding Transformer Architecture

The transformer model is built on an encoder-decoder architecture where both the encoder and decoder are composed of a series of layers that utilize self-attention mechanisms and feed-forward neural networks. This architecture enables the model to process input data in parallel making it highly efficient and effective for tasks involving sequential data.

- The encoder processes input sequences and creates meaningful representations.
- The decoder generates outputs based on encoder representations and previously predicted tokens.
-

The encoder and decoder work together to transform the input into the desired output such as translating a sentence from one language to another or generating a response to a query.



Transformer's Architecture

Key Components Transformer

Transformers have 2 main components:

Encoder

The primary function of the encoder is to create a high-dimensional representation of the input sequence that the decoder can use to generate the output. Encoder consists multiple layers and each layer is composed of two main sub-layers:

> **Self-Attention Mechanism**: This sub-layer allows the encoder to weigh the importance of different parts of the input sequence differently to capture dependencies regardless of their distance within the sequence.
>
> **Feed-Forward Neural Network**: This sub-layer consists of two linear transformations with a ReLU activation in between. It processes the output of the self-attention mechanism to generate a refined representation.

Layer normalization and residual connections are used around each of these sub-layers to ensure stability and improve convergence during training.

Decoder

Decoder in transformer also consists of multiple identical layers. Its primary function is to generate the output sequence based on the representations provided by the encoder and the previously generated tokens of the output.

Each decoder layer consists of three main sub-layers:

Masked Self-Attention Mechanism: Similar to the encoder's self-attention mechanism but with a mask to prevent the decoder from attending to future tokens in the output sequence.

Encoder-Decoder Attention Mechanism: This sub-layer allows the decoder to focus on relevant parts of the encoder's output representation, facilitating the generation of coherent and contextually appropriate output sequences.

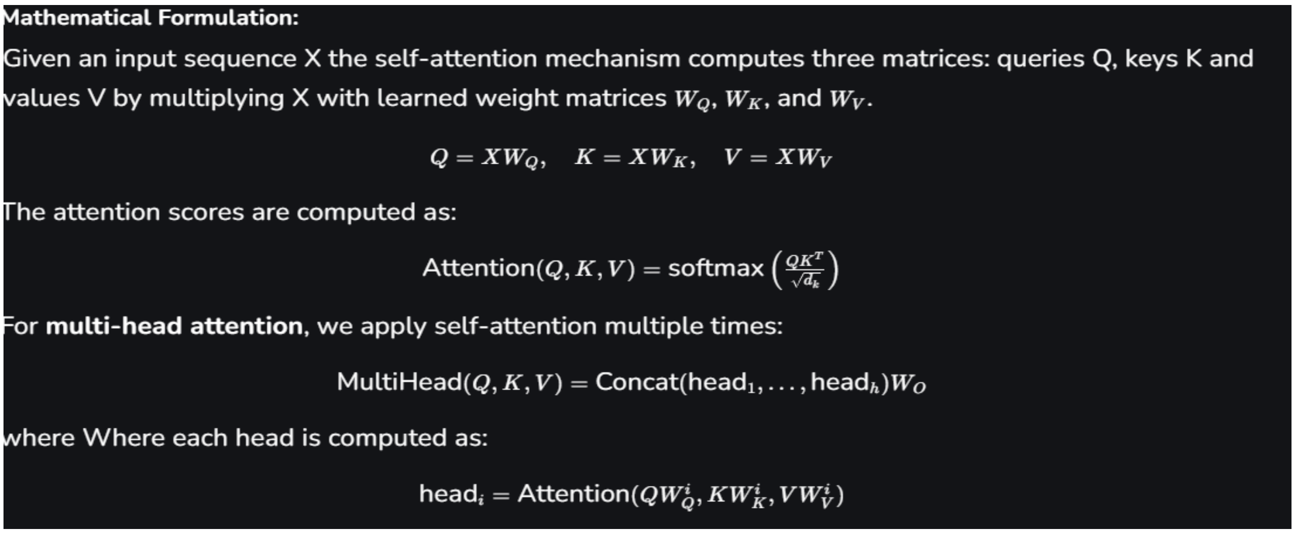
Feed-Forward Neural Network: This sub-layer processes the combined output of the masked self-attention and encoder-decoder attention mechanisms.

In-Depth Analysis of Transformer Components

Multi-Head Self-Attention Mechanism
Multi-head attention extends the self-attention mechanism by applying it multiple times in parallel with each "head" learning different aspects of the input data. This allows the model to capture a richer set of relationships within the input sequence. The outputs of these heads are then concatenated and linearly transformed to produce the final output. The benefits include:

- Improved ability to capture complex patterns in the data.
- Enhanced model capacity without significant increase in computational complexity.

Mathematical Formulation:



Mathematical Formulation:

Given an input sequence X the self-attention mechanism computes three matrices: queries Q, keys K and values V by multiplying X with learned weight matrices W_Q, W_K, and W_V.

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

The attention scores are computed as:

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

For **multi-head attention**, we apply self-attention multiple times:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

where Where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$

Positional Encoding

Transformers lack inherent information about the order of the input sequence due to their parallel processing nature. Positional encoding is introduced to provide the model with information about the position of each token in the sequence.

Positional encodings are added to the input embeddings to give the model a sense of token order. These encodings can be either learned or fixed.

Layer Normalization and Residual Connections

Layer Normalization: stabilizes training by normalizing inputs.

Residual Connections: help avoid vanishing gradients by adding input back to outputs

$$\textbf{Output}=\textbf{LayerNorm}(x+\textbf{SubLayer}(x))$$

This addition helps in preserving the original input information which is crucial for learning complex representations.

How Transformers Work

Input Representation

The first step in processing input data involves converting raw text into a format that the transformer model can understand. This involves tokenization and embedding.

- **Tokenization**: The input text is split into smaller units called tokens, which can be words, subwords, or characters. Tokenization ensures that the text is broken down into manageable pieces.
- **Embedding**: Each token is then converted into a fixed-size vector using an embedding layer. This layer maps each token to a dense vector representation that captures its semantic meaning.
- Positional encodings are added to these embeddings to provide information about the token positions within the sequence.

Encoder Process in Transformers

- **Input Embedding**: The input sequence is tokenized and converted into embeddings with positional encodings added.
- **Self-Attention Mechanism**: Each token in the input sequence attends to every other token to capture dependencies and contextual information.
- **Feed-Forward Network**: The output from the self-attention mechanism is passed through a position-wise feed-forward network.
- **Layer Normalization and Residual Connections**: Layer normalization and residual connections are applied.

Decoder Process

- **Input Embedding and Positional Encoding**: The partially generated output sequence is tokenized and embedded with positional encodings added.
- **Masked Self-Attention Mechanism**: The decoder uses masked self-attention to prevent attending to future tokens ensuring that the model generates the sequence step-by-step.
- **Encoder-Decoder Attention Mechanism**: The decoder attends to the encoder's output allowing it to focus on relevant parts of the input sequence.
- **Feed-Forward Network**: Similar to the encoder the output from the attention mechanisms is passed through a position-wise feed-forward network.
- **Layer Normalization and Residual Connections**: Similar to the encoder Layer normalization and residual connections are applied.

Training and Inference

Transformers are trained using supervised learning where the model learns to predict the next token in a sequence given the previous tokens.

Transformers have transformed deep learning by using self-attention mechanisms to efficiently process and generate sequences capturing long-range dependencies and contextual relationships. Their encoder-decoder architecture combined with multi-head attention and feed-forward networks enables highly effective handling of sequential data.