Transformers

What is a Transformer?

A Transformer is a deep learning model architecture introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017. The key innovation of transformers is the self-attention mechanism, which allows the model to process all elements in a sequence simultaneously, rather than one at a time like older models.

In simpler terms, a Transformer is a type of neural network that excels in tasks involving sequential data, such as language modeling, translation, and text generation. The transformer architecture has two main parts:

- **Encoder**: Processes the input sequence.
- **Decoder**: Generates the output sequence.

Key Components of a Transformer

The Transformer is composed of multi-head self-attention and position-wise feed-forward networks.

1. Attention Mechanism

The most crucial part of the Transformer is the attention mechanism, specifically self-attention.

- Self-Attention: This mechanism allows each word (token) in the input to attend to every other word in the sequence. For each word, the model computes how much focus (or “attention”) it should give to all other words in the sequence. The result is a weighted sum of these words, which is used to represent each word’s context.

2. Multi-Head Attention

Rather than computing just one set of attention scores, the Transformer computes multiple sets (called "heads"). This allows the model to focus on

different aspects of the input sequence in parallel. The outputs from each head are concatenated and passed through a linear layer.

3. Positional Encoding

Since transformers process all words simultaneously (in parallel), they don't have a natural sense of order (unlike RNNs or LSTMs). To handle this, positional encodings are added to the input embeddings, giving the model information about the position of each word in the sequence.

This is achieved using sine and cosine functions with different frequencies to encode the position.

4. Feed-Forward Neural Networks

After the attention mechanism, each word representation is passed through a position-wise feed-forward network. This is a simple, fully connected network applied to each token independently.

The feed-forward network consists of two layers:

- A dense layer (with ReLU activation) to project the word representation to a higher-dimensional space.
- Another dense layer to project it back to the original space.

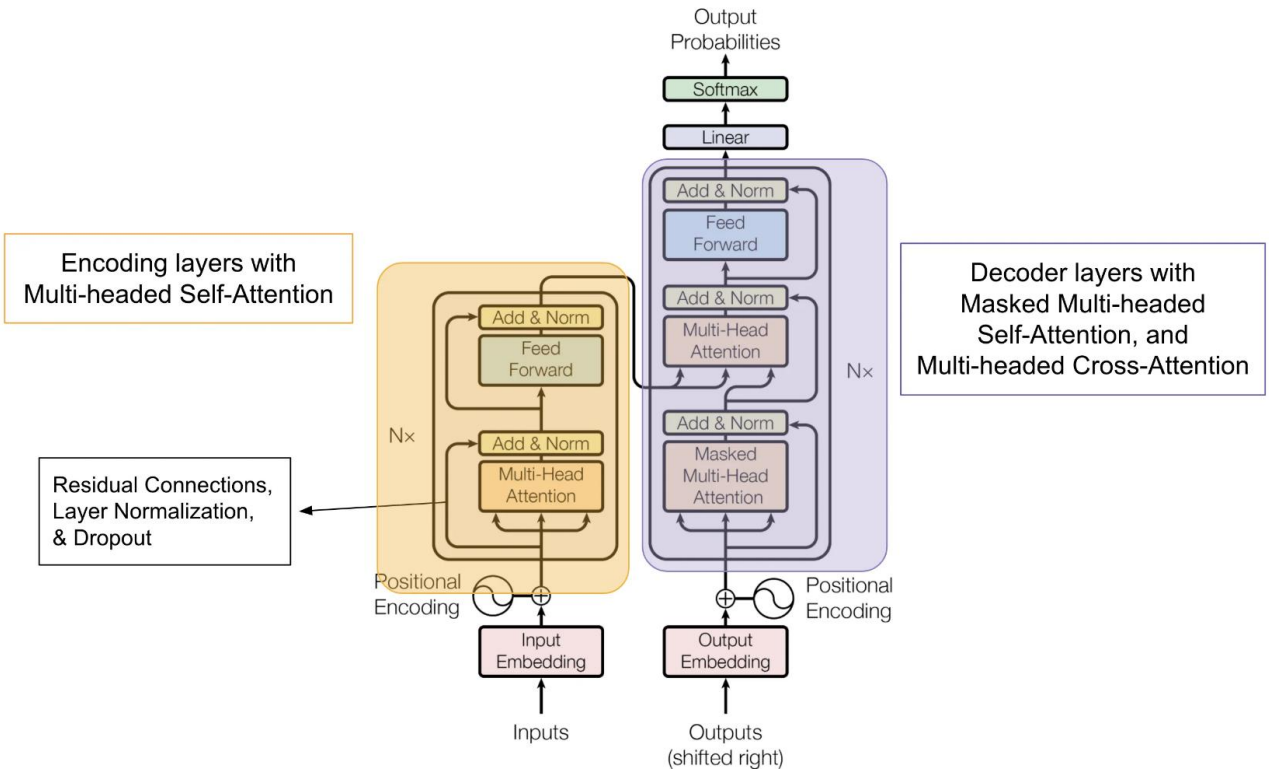
Transformer Architecture: Encoder and Decoder

1. Encoder

The encoder is responsible for processing the input sequence. It consists of multiple layers, and each layer has two key components:

- Multi-Head Self-Attention: The self-attention mechanism helps the encoder pay attention to all words in the sequence.
- Position-wise Feed-Forward Network: After the attention operation, the result is passed through a feed-forward neural network.

Each encoder layer also includes layer normalization and residual connections, which help improve training stability.



2. Decoder

The decoder generates the output sequence. It has a similar structure to the encoder, but it also has an additional component:

- Masked Multi-Head Self-Attention: In the decoder, self-attention is masked to ensure that the decoder only attends to previous words in the sequence, preventing the model from "seeing the future."
- Encoder-Decoder Attention: The decoder also attends to the encoder's output (encoded representations of the input sequence), providing a mechanism for generating meaningful output based on the input.

Like the encoder, the decoder includes feed-forward layers and layer normalization.

Transformer Variants

Since the introduction of the original transformer, many variants and models have been developed that have improved performance on a variety of tasks. Some key variants include:

1. BERT (Bidirectional Encoder Representations from Transformers)

- Type: Encoder-only transformer.
- Key Feature: BERT uses a masked language modeling approach to pre-train the model. It looks at the entire input (bidirectionally) and predicts missing words in a sentence.
- Use Cases: Sentiment analysis, question answering, named entity recognition (NER).

2. GPT (Generative Pre-trained Transformer)

- Type: Decoder-only transformer.
- Key Feature: GPT uses a causal (unidirectional) language modeling approach, predicting the next word in a sequence.
- Use Cases: Text generation, language modeling, code completion.

3. T5 (Text-to-Text Transfer Transformer)

- Type: Encoder-decoder transformer.
- Key Feature: T5 frames all NLP tasks as text-to-text problems, where both input and output are text sequences. This unification allows the model to be fine-tuned for various tasks like summarization, translation, etc.
- Use Cases: Text summarization, translation, text classification.

4. Vision Transformers (ViT)

- Type: Transformer applied to vision tasks.
- Key Feature: Vision Transformers apply the same transformer architecture to image patches, treating image patches as tokens to be processed by the self-attention mechanism.
- Use Cases: Image classification, object detection, segmentation.

Why Transformers Are So Powerful

The Transformer architecture has revolutionized NLP and other domains for several reasons:

1. Parallelization

Unlike RNNs or LSTMs, which process data sequentially (one token at a time), transformers process all tokens simultaneously. This parallelism leads to **faster training** and allows transformers to scale to large datasets.

2. Long-Range Dependencies

Transformers can capture long-range dependencies in the data more effectively. The self-attention mechanism allows each token to attend to every other token, making it easier to capture relationships between distant words in a sequence.

3. Scalability

Transformers scale well with large datasets and large models, enabling the creation of highly accurate models like GPT-3 and BERT-large. Transformers can handle millions (or even billions) of parameters efficiently.

4. Transfer Learning

Transformers, especially with pre-trained models like BERT and GPT, excel at transfer learning. Once a transformer model is pre-trained on a large corpus, it can be fine-tuned on smaller, task-specific datasets, achieving state-of-the-art performance on a wide variety of tasks.