Transformers

1. Introduction

Transformers are a type of deep learning architecture introduced by Vaswani et al. in the seminal paper **“Attention is All You Need” (2017)**. They are designed to handle sequential data, primarily in **natural language processing (NLP)** tasks such as language translation, text summarization, and question answering. Unlike previous sequence models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), Transformers do not rely on recurrence or convolutions. Instead, they use **self-attention** mechanisms to process the entire input sequence in parallel.

---

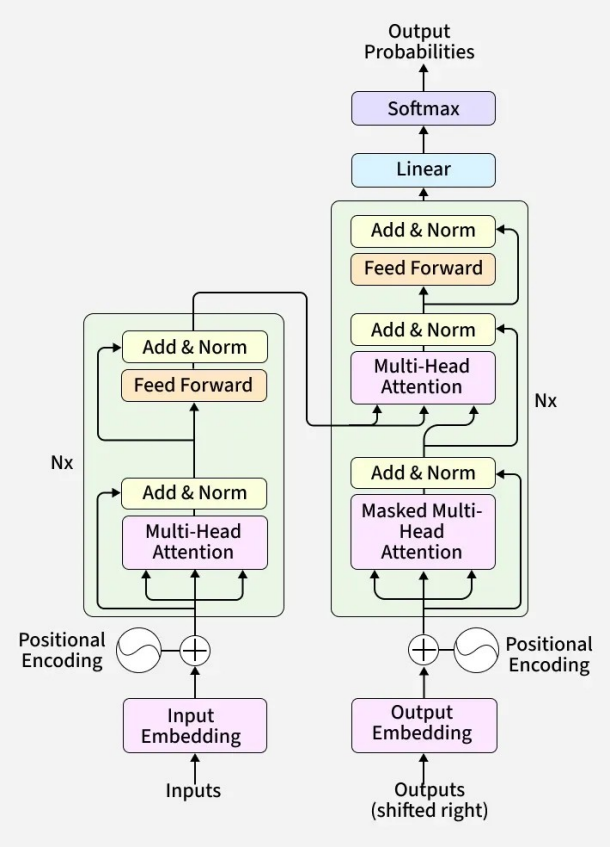
2. Transformer Architecture Overview

The Transformer model consists of two main parts:

- **Encoder:** Converts the input sequence into a continuous representation.
- **Decoder:** Uses the encoder output and previously generated tokens to produce the output sequence.

Both encoder and decoder are composed of **N identical layers** (typically 6), each containing two main sub-layers:

1. **Multi-Head Self-Attention Mechanism**
2. **Position-wise Feed-Forward Neural Network**



2.1 Input Embedding and Positional Encoding

Since Transformers lack recurrence, they have no built-in sense of the order of tokens. To solve this, input tokens (words or subwords) are first converted to dense vectors via an **embedding layer**. Then, **positional encodings** are added to the embeddings to inject information about the token positions.

- **Positional Encoding Formula:**

For each position pos and dimension i:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

where d_{model} is the embedding size.

Example:
For the sentence *"Transformers process sequences"*, each word is embedded and combined with a positional encoding to preserve order information.

---

3. Self-Attention Mechanism

3.1 Concept

Self-attention allows the model to weigh the importance of other tokens in the sequence when encoding a specific token. This is critical for capturing dependencies regardless of their distance in the sequence.

3.2 Computation

For each token in the sequence, three vectors are computed:

- **Query (Q)**
- **Key (K)**
- **Value (V)**

These vectors are derived by multiplying the input embeddings by learned weight matrices

The attention scores are computed by taking the dot product between the Query vector of the current token and the Key vectors of all tokens, scaling them, and applying a softmax to get attention weights.

3.3 Example

Consider the sentence: *"The animal didn't cross the street because it was too tired."*

When encoding the word *"it"*, the self-attention mechanism helps the model focus on the word *"animal"* rather than other less relevant words, because *"it"* refers to *"animal"*.

---

4. Multi-Head Attention

Instead of performing a single attention calculation, Transformers use **multiple parallel attention heads**. Each head independently computes attention using different learned projections of Q, K, and V, allowing the model to capture diverse types of relationships.

The outputs of all heads are concatenated and projected to the final vector.

Example:

- Head 1 might focus on syntactic relations (like subject-verb agreement).
- Head 2 might focus on semantic meaning (like co-references).

---

5. Position-Wise Feed-Forward Networks

Each encoder and decoder layer contains a fully connected feed-forward network applied independently to each position. This network consists of two linear transformations with a ReLU activation in between:

---

6. Residual Connections and Layer Normalization

To improve gradient flow and training stability, each sub-layer (attention and feed-forward) uses a **residual connection** followed by **layer normalization**:

---

7. Encoder and Decoder Details

7.1 Encoder

The encoder takes the input sequence embeddings plus positional encodings and passes them through a stack of identical layers, each performing multi-head self-attention and feed-forward processing.

7.2 Decoder

The decoder is similar but has an additional **masked multi-head self-attention** layer that prevents positions from attending to subsequent tokens during training, ensuring autoregressive generation.

It also performs **encoder-decoder attention**, where the decoder attends to the encoder output, integrating the encoded input context.

---

8. Example Application: Neural Machine Translation

Suppose we want to translate the English sentence:

“The quick brown fox jumps over the lazy dog.”

- **Encoding:** Each word is embedded, combined with positional encodings, and processed through the encoder layers. Self-attention helps the model understand context, such as the subject “fox” and action “jumps.”
- **Decoding:** The decoder generates the French translation step-by-step, using masked self-attention to prevent cheating by looking at future words, and encoder-decoder attention to reference the English sentence’s context.

Example output:

“Le renard brun rapide saute par-dessus le chien paresseux.”

---

9. Advantages Over Previous Models

- **Parallelization:** Unlike RNNs which process sequentially, Transformers process entire sequences at once, significantly reducing training time.
- **Long-Range Dependency Handling:** Self-attention captures relationships between distant words better than RNNs or CNNs.
- **Scalability:** Transformers can be scaled up to very large models (e.g., GPT-4, BERT), enabling state-of-the-art performance.

What is a Transformer?

Imagine you want to read a story and then tell your friend what it’s about. To do that well, you need to remember important parts of the story and understand how they connect.

A **Transformer** is like a smart robot brain that reads sentences and figures out which words are important and how they relate to each other — even if they’re far apart in the sentence!

How Does It Work?

1. **Breaking down the sentence:**
   The Transformer first breaks a sentence into smaller pieces called **tokens** (usually words).
2. **Remembering word order:**
   Since the Transformer looks at all words at once, it needs to know which word comes first, second, and so on. It adds a special code called **positional encoding** to each word to remember this.
3. **Paying attention:**
   When the Transformer looks at a word, it asks,
   "Which other words should I pay attention to so I can understand this word better?"
   This is called **self-attention**.
4. **Putting it all together:**
   After figuring out all the relationships between words, the Transformer can do many tasks, like translating languages, summarizing stories, or answering questions.