Transformers

YOUTUBEVIDEO : https://www.youtube.com/watch?v=TQQlZhbC5ps&t=15s

---

What Are Transformers?

Transformers are a type of neural network architecture designed specifically for processing sequential data (like text), without using recurrence like RNNs. Instead, they rely entirely on self-attention mechanisms to understand relationships between all words in a sentence simultaneously. This allows them to process entire sequences in parallel, making them efficient and scalable.

Why Use Transformers?

- **No recurrence** = Faster parallel processing
- **Self-attention** = Better understanding of global context
- **Scalability** = Works well with long sequences

Transformers are the foundation of modern NLP models like BERT, GPT, T5, and others.

Step-by-Step Transformer Architecture

1. **Input Tokenization**
   Input: "I love you"
   Tokenize into: ["I", "love", "you"]
   Convert to token IDs: [101, 2203, 2017] (example IDs)
2. **Word Embedding + Positional Encoding**

- Word Embedding: Converts token IDs into dense vectors
- Positional Encoding: Injects information about word position (since Transformer has no sense of order on its own)
  Result: Each word gets a vector combining meaning and position

3. **Encoder Block**
   Each encoder block contains:

- Multi-Head Self-Attention: Each word attends to every other word to understand context
- Feedforward Layer: Transforms the self-attended vectors

- Add & Normalize: Residual connection + layer normalization
  Multiple encoder blocks are stacked for depth

4. **Decoder Block (used in translation)**
   Each decoder block contains:

- Masked Self-Attention: Prevents looking ahead while generating output
- Encoder-Decoder Attention: Connects encoder output with decoder input
- Feedforward Layer: Transforms decoder representations

5. **Final Output**

- The decoder produces vectors representing each output word.
- These vectors pass through a **Linear layer**, which maps the high-dimensional decoder output into a vector the size of the vocabulary.
- The **Softmax layer** then converts these scores into probabilities for each word in the vocabulary.
- The model predicts the word with the highest probability and generates output tokens one-by-one until it produces an end-of-sequence token.

Summary Flow

Input Text → Token IDs → Embedding + Position → Encoder (Self-Attention + Feedforward) → Decoder → Linear layer → Softmax layer → Output Words