Translation, Attention, Transfomer

1.Bài toán dịch máy(Machine Translation)

- **Translation (dịch tự động)** là nhiệm vụ biến một câu (hoặc đoạn văn) trong ngôn ngữ nguồn thành câu (hoặc đoạn văn) tương đương trong ngôn ngữ đích, giữ nội dung nghĩa, ngữ pháp, ngữ cảnh.

- Trước khi các mô hình deep learning phát triển, người ta dùng các hệ thống dựa trên quy tắc (rule-based), sau đó hệ thống thống kê (statistical MT, như phrase-based).

- Với deep learning, ta tiếp cận bằng các mô hình học chuỗi → chuỗi (sequence-to-sequence), và sau đó thêm attention để cải thiện hiệu quả.

2.Mô hình Chuỗi-sang-Chuỗi (Sequence-to-Sequence Models - Seq2Seq)

2.1. Ý tưởng cơ bản

Mô hình Seq2Seq biến một chuỗi đầu vào (ví dụ: từ, ký tự) thành một chuỗi đầu ra.

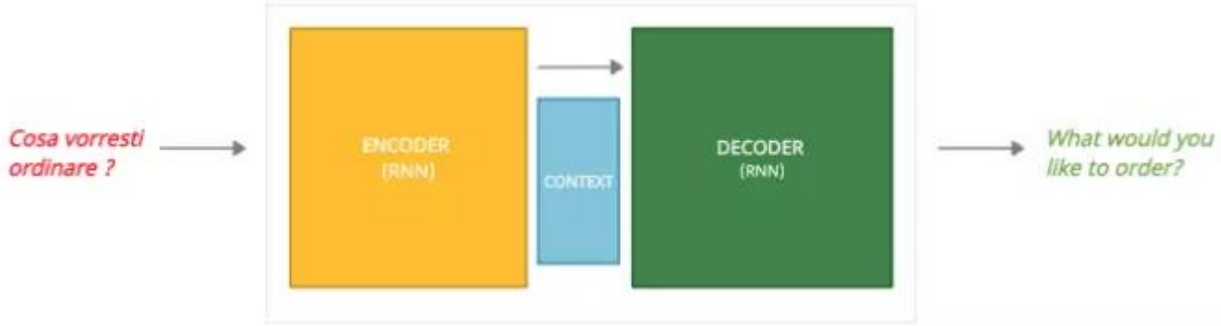
Kiến trúc chính của mô hình Seq2Seq bao gồm hai mạng nơ-ron:

Encoder (Mã hóa):

- Đọc và xử lý câu nguồn (input sequence) từng từ một (thường là dùng **LSTM** hoặc **RNN**).
- Đầu ra của Encoder là một **hidden state cuối cùng (final hidden state)**, đóng vai trò là một véc-tơ tóm tắt, mã hóa toàn bộ thông tin của câu nguồn.

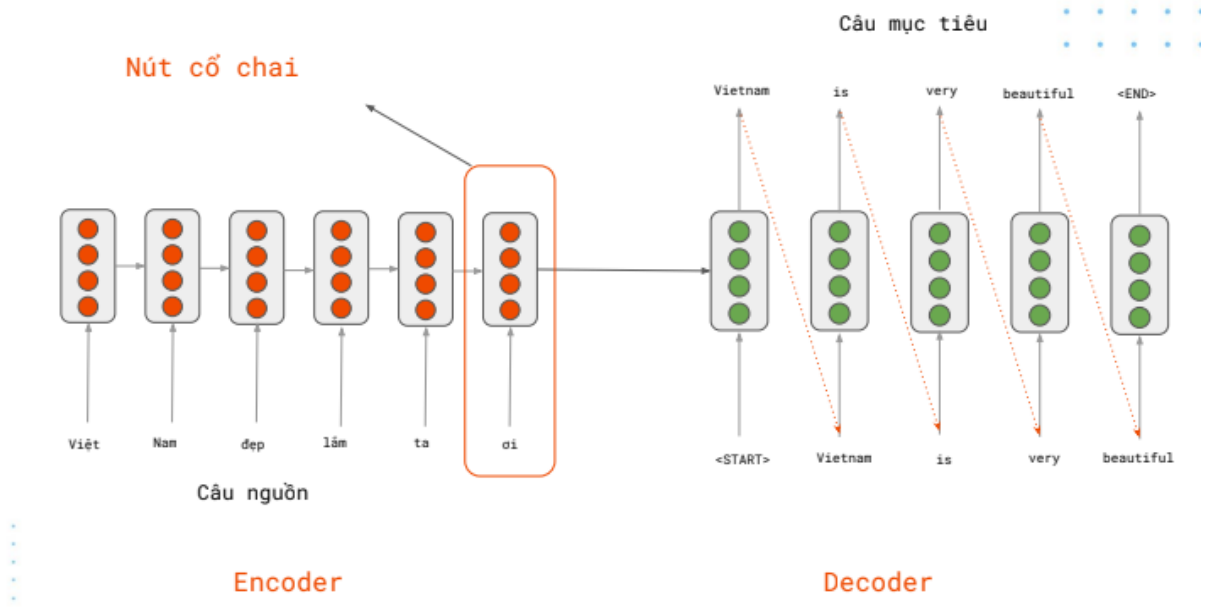
Decoder (Giải mã):

- Là một mô hình ngôn ngữ có điều kiện, nhận **hidden state cuối cùng** từ Encoder làm trạng thái ẩn ban đầu.
- Tạo ra (sinh ra) câu dịch (output sequence) từng từ một, dựa trên hidden state được truyền vào và các từ đã được sinh ra trước đó .



2.2. Hạn chế của mô hình Seq2Seq thuần túy

- **Bottleneck (nút cổ chai)**: việc ép cả thông tin của chuỗi nguồn dài (có thể rất dài) vào một vector cố định (hidden vector) là rất hạn chế. Khi câu nguồn quá dài hoặc phức tạp, vector ẩn không thể “nhét” đủ thông tin.



- **Khó khăn với từ hoặc cụm từ quan trọng**: mô hình có thể quên hoặc bỏ sót thông tin quan trọng, đặc biệt khi chuỗi nguồn dài hoặc có nhiều chi tiết ngữ cảnh.

- **Dịch không mượt, sai ngữ cảnh** nếu chỉ dựa vào vector cố định duy nhất.

3. Attention

3.1. Ý tưởng

Attention cung cấp một giải pháp cho vấn đề *bottleneck* .

Ý tưởng cốt lõi của Attention:

- Thay vì sử dụng 1 context vector duy nhất (last state) khi dịch tất cả các từ, ta sử dụng mỗi context vector riêng biệt khi predict ra từng từ.
- Mỗi context vector được tổng hợp có trọng số từ tất cả các state trong encode.

Với query Q, keys K, và values V :

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

3.2. Cơ chế Attention

Giả sử:

-Bộ mã hóa (Encoder) tạo ra các state ẩn(tương ứng với từng từ trong câu đầu vào):

$$h_1, h_2, ..., h_N \in \mathbb{R}^h$$

-Tại bước t của Decoder, ta có state hiện tại:

$$\boldsymbol{s}_t \in \mathbb{R}^h$$

1. Tính điểm Attention (alignment scores):

Với mỗi state hi của Encoder, ta đo độ liên quan giữa st và hi:

$$e_t = [s_t^T h_1,\ s_t^T h_2,\ ...,\ s_t^T h_N] \in \mathbb{R}^N$$

2. Chuẩn hóa thành phân phối trọng số Attention:

$$\alpha_t = \text{softmax}(e_t) \in \mathbb{R}^N$$

Mỗi α_t^i thể hiện **mức độ mô hình tập trung vào từ thứ i trong input** ở bước dịch hiện tại.

3. Tạo vector ngữ cảnh có trọng số (Context Vector):

$$a_t = \sum_{i=1}^{N} \alpha_t^i \cdot h_i \in \mathbb{R}^h$$

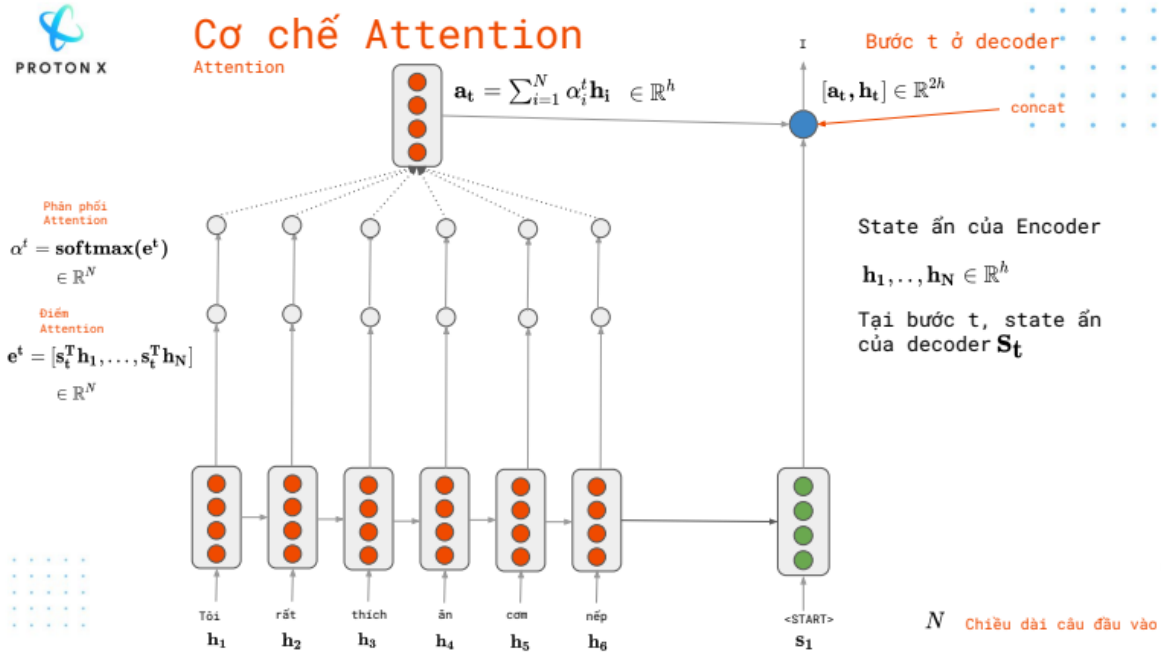
Vector này là bản tóm tắt động của câu đầu vào – nhưng chỉ nhấn mạnh vào những từ liên quan nhất đến bước dịch hiện tại

4. Kết hợp với state Decoder để sinh từ tiếp theo:

Ta **nối (concat)** context vector a_t với state hiện tại s_t:

$$[a_t,\ s_t] \in \mathbb{R}^{2h}$$

Vector kết hợp này sẽ được đưa vào layer dự đoán để sinh ra từ tiếp theo trong chuỗi output.



Nó giải quyết hoàn toàn nhược điểm **"bottleneck"** của mô hình Seq2Seq truyền thống, đồng thời giúp kết quả dịch chính xác hơn, đặc biệt với **các câu dài hoặc có cấu trúc phức tạp**.

4. Self-attention

4.1.Self-attention là gì

Nếu Attention trong Seq2Seq giúp decoder tập trung vào các từ quan trọng của encoder, thì Self-Attention làm điều tương tự nhưng ngay bên trong một chuỗi.

Self-Attention không chỉ nhìn *xung quanh* mà còn *nhìn chính mình*. Nhờ đó:

- Giữ lại nghĩa của chính từ đó (không bị trộn lẫn).
- Phân biệt các từ giống nhau nhưng khác vai trò ngữ pháp.

4.2. Self-attention trong transfomer

Self-Attention là một cơ chế cốt lõi trong kiến trúc Transformer, nó cho phép mô hình xử lý các mối quan hệ phức tạp giữa các phần tử trong chuỗi dữ liệu (như từ ngữ trong câu) một cách hiệu quả, mà không cần đến các lớp recurrent như RNN hay LSTM. Self-Attention giúp mô hình tập trung vào các thông tin quan trọng trong ngữ cảnh, giải quyết hạn chế của word embedding đơn lẻ – nơi ý nghĩa của từ có thể thay đổi tùy theo vị trí trong câu.

Nguyên Lý Hoạt Động

Kiến trúc Transformer bao gồm một lớp self-attention, nơi mà toàn bộ quá trình attention được tích hợp. Các bước được trình bày theo nghiên cứu gốc của Ashish Vaswani và cộng sự trong bài báo “Attention is All You Need”:

1. **Mã hóa chuỗi đầu vào:** Chuyển token (từ hoặc ký tự) thành vector embedding để biểu diễn ngữ nghĩa, dễ dàng tính toán attention (ví dụ: trong dịch máy, một câu trở thành chuỗi embedding).
2. **Tạo vector Q, K, V:** Áp dụng biến đổi tuyến tính (linear transformation) trên embedding để tạo ba vector cho mỗi token: Query (Q – truy vấn), Key (K – khóa), và Value (V – giá trị). Những vector này giúp xác định mối liên kết ngữ nghĩa.

3. **Tính điểm attention:** Sử dụng tích vô hướng (dot product) giữa Q và K, sau đó chia cho (kích thước key) để ổn định gradient và tránh giá trị lớn.
4. **Chuyển thành xác suất**: Áp dụng hàm softmax trên điểm attention để chuẩn hóa thành trọng số (phân phối xác suất), đánh giá tầm quan trọng tương đối của từng phần tử.
5. **Tổng có trọng số:** Nhân trọng số với V để tạo output – một tổng hợp tập trung, nơi các phần tử quan trọng ảnh hưởng mạnh hơn đến kết quả cuối cùng.

Quá trình này cho phép mô hình xử lý toàn bộ chuỗi song song, tạo ra biểu diễn ngữ cảnh phong phú.

4.3 Lợi Ích Vượt Trội của Self-Attention

Self-Attention mang lại nhiều ưu điểm so với các phương pháp truyền thống:

- Phụ thuộc dài hạn và hiểu ngữ cảnh: Nhận biết mối quan hệ xa, phân bổ trọng số phù hợp dựa trên ngữ cảnh tổng thể.
- Tính toán song song và hiệu quả: Xử lý đồng thời các phần tử, độ phức tạp tuyến tính theo độ dài chuỗi, tăng tốc huấn luyện trên dữ liệu lớn.
- Khả năng mở rộng và linh hoạt: Dễ dàng điều chỉnh cho chuỗi dài biến đổi, học biểu diễn phức tạp mà không cần recurrent layers.

- Học hỏi mạnh mẽ: Tự động trích xuất đặc trưng ý nghĩa, cải thiện chất lượng mô hình hóa phụ thuộc xa và tổng quát hóa tốt.

Những lợi ích này làm cho Self-Attention trở thành backbone cho các mô hình lớn, giảm thời gian huấn luyện và nâng cao hiệu suất.

5. Transformer

5.1. Giới thiệu

Transformer là một mô hình học sâu được giới thiệu trong bài báo "Attention is All You Need" (Vaswani et al., 2017), đánh dấu sự thay đổi lớn trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Mô hình này loại bỏ hoàn toàn kiến trúc Seq2Seq truyền thống dựa trên RNN/LSTM, thay vào đó sử dụng cơ chế Attention làm lõi chính. Transformer cho phép xử lý song song hiệu quả, khai thác tốt các phụ thuộc dài hạn trong dữ liệu chuỗi, và đạt hiệu suất cao hơn trên các nhiệm vụ như dịch máy.

Các đặc điểm chính:

- Loại bỏ Seq2Seq: Không sử dụng các lớp recurrent (RNN), giúp dễ dàng song song hóa.
- Attention làm core: Toàn bộ mô hình dựa trên self-attention và multi-head attention.
- Hàm mất mát: Sử dụng Cross-Entropy loss, tương tự các mô hình seq2seq cổ điển.

Kiến trúc Transformer bao gồm hai phần chính: Encoder và Decoder, mỗi phần gồm nhiều layer chồng lên nhau.

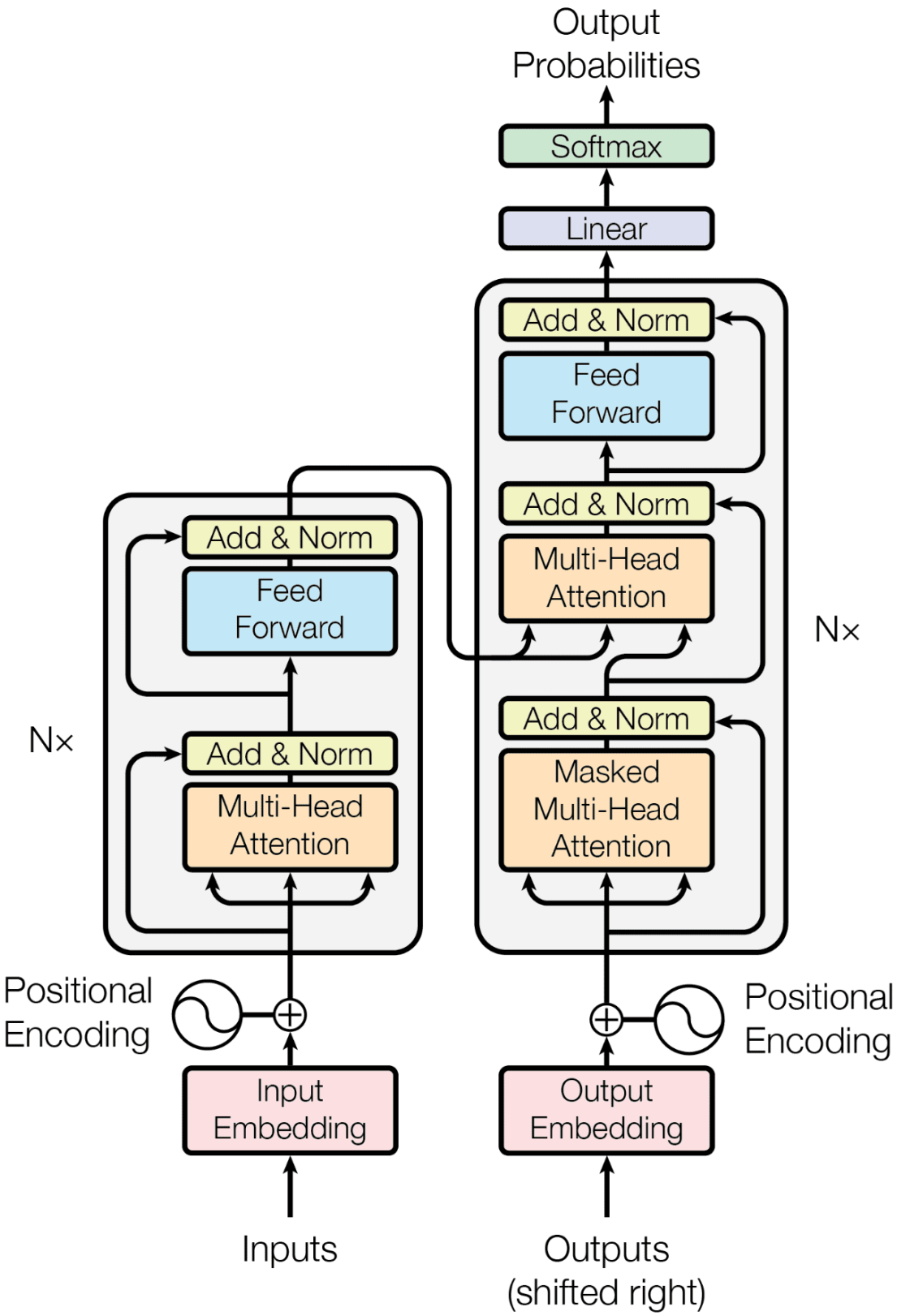
5.2. Kiến trúc tổng quát

Transformer bao gồm một stack của N layer Encoder và N layer Decoder (thường N=6). Đầu vào và đầu ra là các chuỗi token được embedding thành vector.

- Input Embedding: Chuyển đổi token thành vector dense, sau đó thêm positional encoding để giữ thông tin vị trí (vì không có recurrent).
- Output: Decoder sản sinh xác suất cho token tiếp theo qua linear layer và softmax.

Sơ đồ kiến trúc cơ bản:

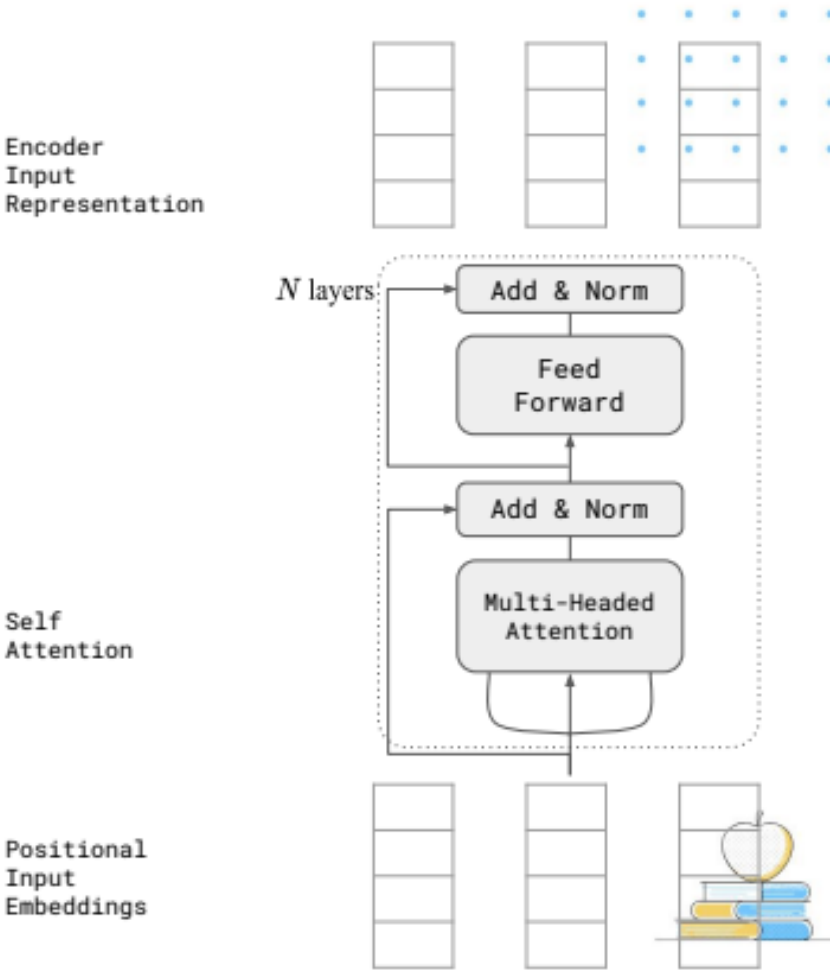
- Encoder: Xử lý input sequence.
- Decoder: Xử lý output sequence (shifted right để tránh nhìn trước), sử dụng attention đến output của encoder.


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

5.3. Encoder

Encoder gồm N layer giống nhau, mỗi layer có hai sub-layer chính:

- Multi-Head Self-Attention: Tính attention giữa tất cả các vị trí trong input sequence, cho phép mô hình tập trung vào các phần liên quan.
- Feed Forward Network (FFN): Một mạng fully-connected hai lớp với ReLU activation, áp dụng độc lập cho từng vị trí.



Mỗi sub-layer được theo sau bởi:

- Residual Connection: Add input của sub-layer vào output (x + sublayer(x)).

- Layer Normalization: Normalize để ổn định training.

5.4. Decoder

Decoder tương tự Encoder nhưng có ba sub-layer:

- Masked Multi-Head Self-Attention: Attention trên output sequence, masked để tránh nhìn token tương lai (autoregressive).
- Multi-Head Attention: Attention từ decoder đến encoder output (cross-attention).
- Feed Forward Network: Giống encoder.

Cũng sử dụng residual connections và layer norm sau mỗi sub-layer.

6. Pre-training

6.1.Pre-training là gì?

Pre-training (tiền huấn luyện) là giai đoạn huấn luyện một mô hình trên **một lượng dữ liệu văn bản cực lớn và không gắn nhãn**, trước khi sử dụng mô hình đó cho một **nhiệm vụ cụ thể** (vd: phân loại cảm xúc, trả lời câu hỏi, dịch máy…).

Thay vì **huấn luyện trực tiếp từ đầu (training from scratch)** cho từng bài toán riêng lẻ — vốn tốn nhiều dữ liệu và tài nguyên — ta huấn luyện một **mô hình nền (foundation model / language model)** để học **kiến thức ngôn ngữ tổng quát**. Sau đó, ta **fine-tune (tinh chỉnh)** mô hình này cho các tác vụ cụ thể.

6.2.Ba cách tiền huấn luyện mô hình (Model Pre-training Three Ways)

Kiến trúc	Tiêu biểu	Cách huấn luyện	Khả năng
Encoder-only	BERT, RoBERTa	Masked Language Modeling (MLM)	Hiểu ngữ cảnh, phân loại
Decoder-only	GPT-1/2/3	Dự đoán token tiếp theo (LM)	Sinh văn bản, in-context learning
Encoder-Decoder	T5, BART	Span Corruption	Cả hiểu và sinh