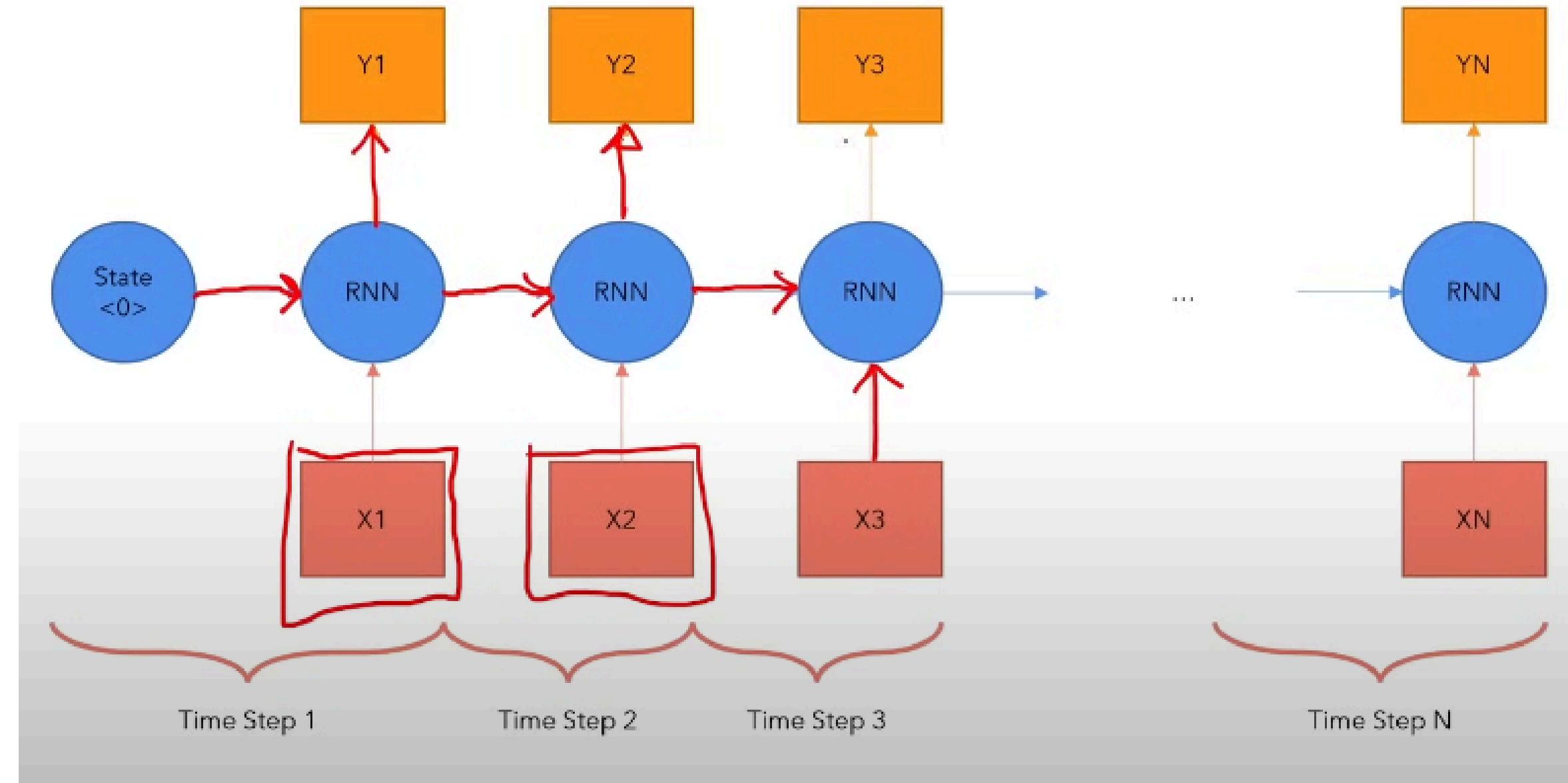


# **TRANSFORMER**

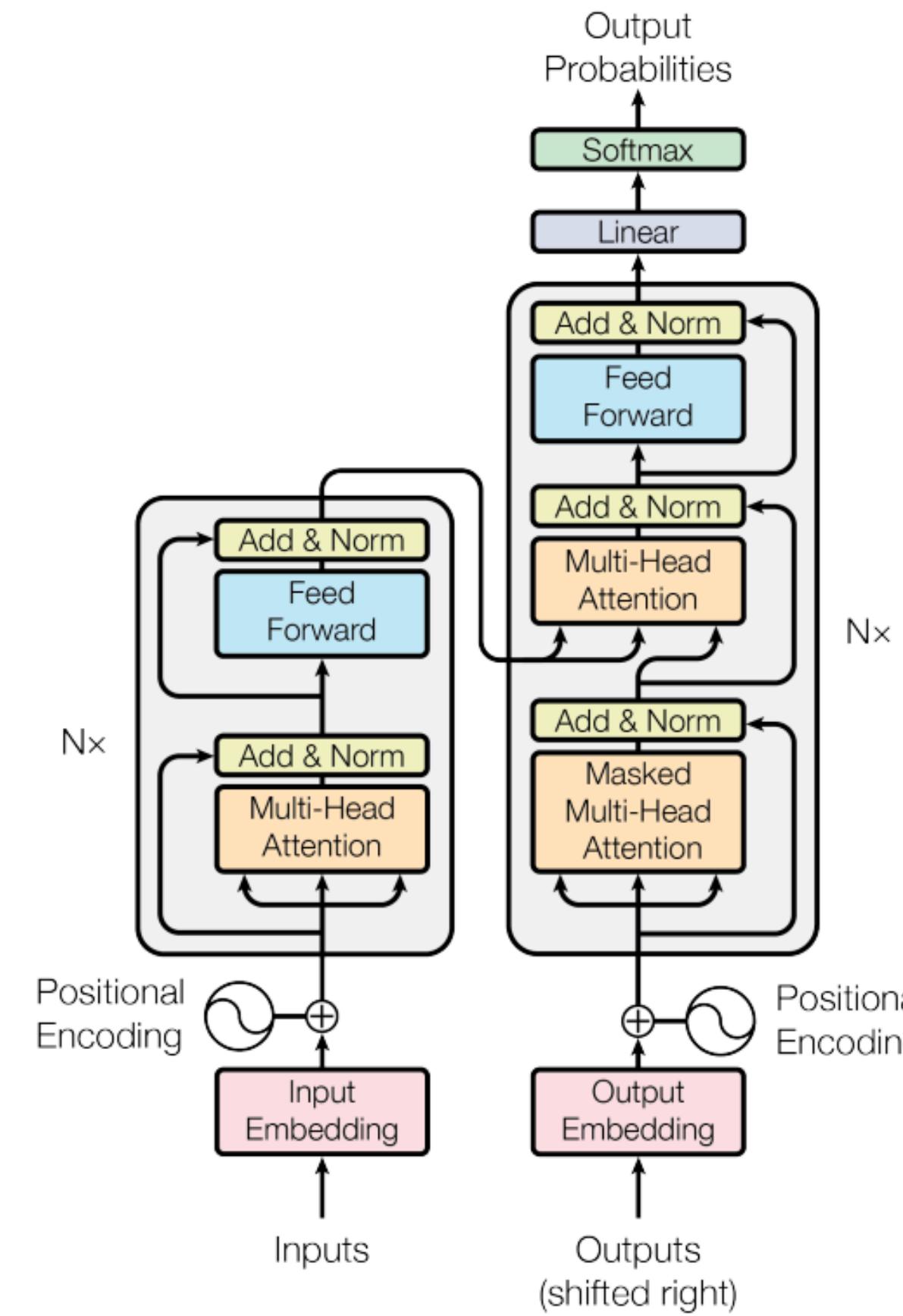
# 1. Hạn chế các mô hình xử lý chuỗi trước đó



- Tính toán tuần tự -> chậm
- Gradient vanishing hoặc exploding
- Khó khăn khi cần ghi nhớ và xử lý thông tin từ các bước thời gian xa trước đó (long-term dependencies)

## 2. Mô hình Transformer

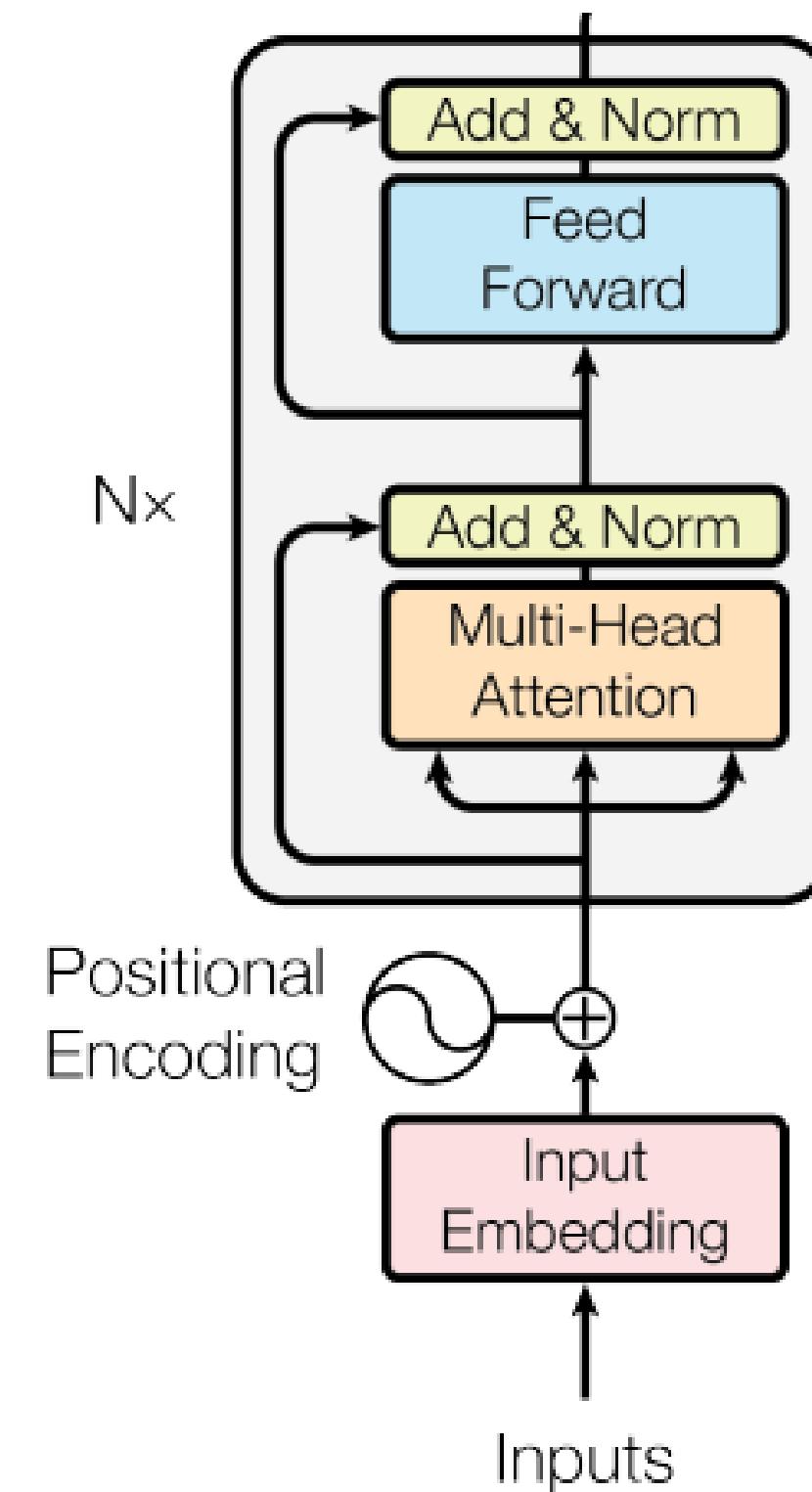
**Khối Encoder** giúp tập trung vào việc hiểu ngữ cảnh của chuỗi đầu vào.



**Khối Decoder** sử dụng thông tin từ encoder để tạo ra các dự đoán tuần tự.

## 2. Mô hình Transformer

### Encoder

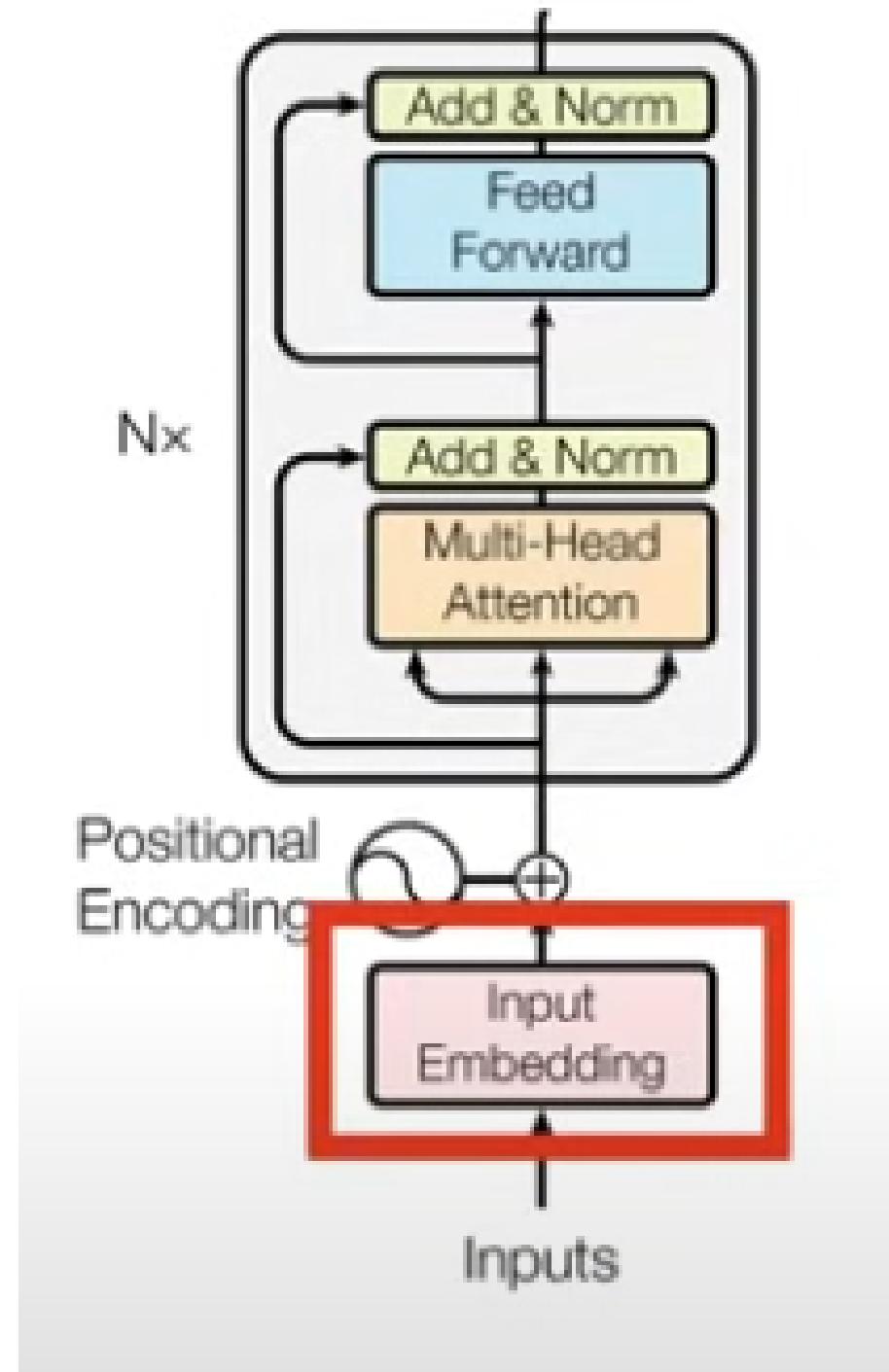


## 2. Mô hình Transformer

### Encoder

#### Input Embedding

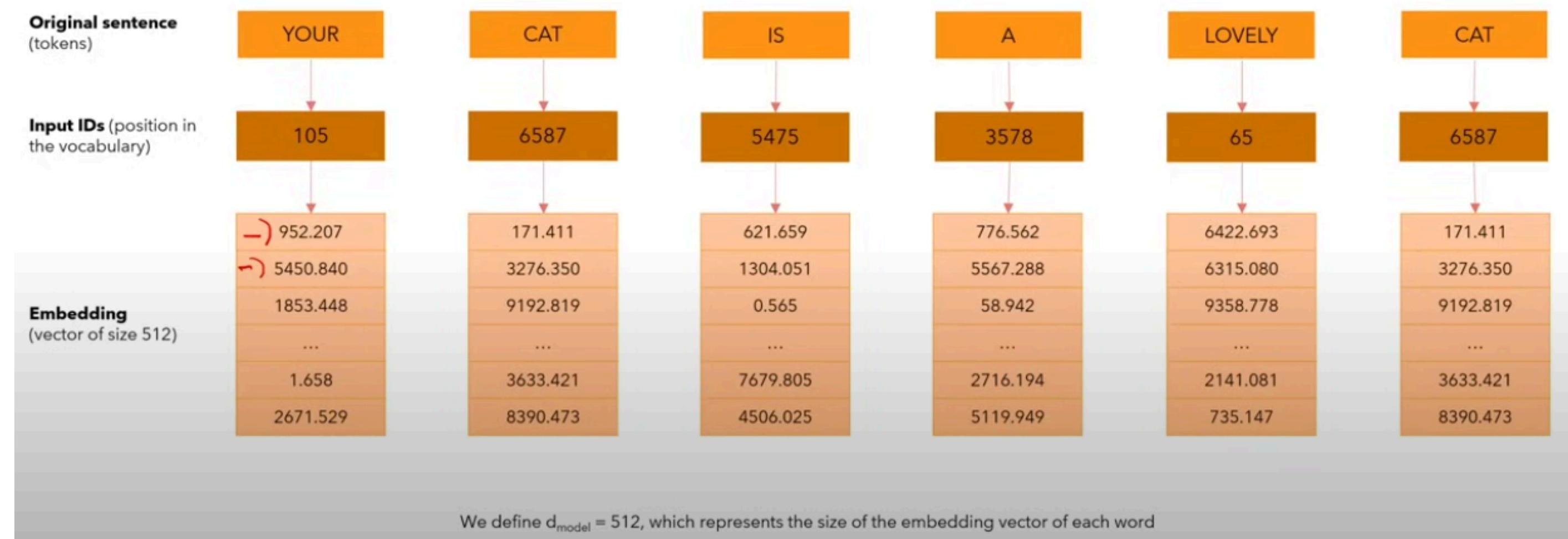
- Biến đổi từ ngữ thành dạng số học để máy tính xử lý được.
- Nắm bắt ngữ nghĩa và ngữ cảnh của từ.
- Giảm chiều dữ liệu, tăng hiệu quả tính toán.



## 2. Mô hình Transformer

### Encoder

#### Input Embedding

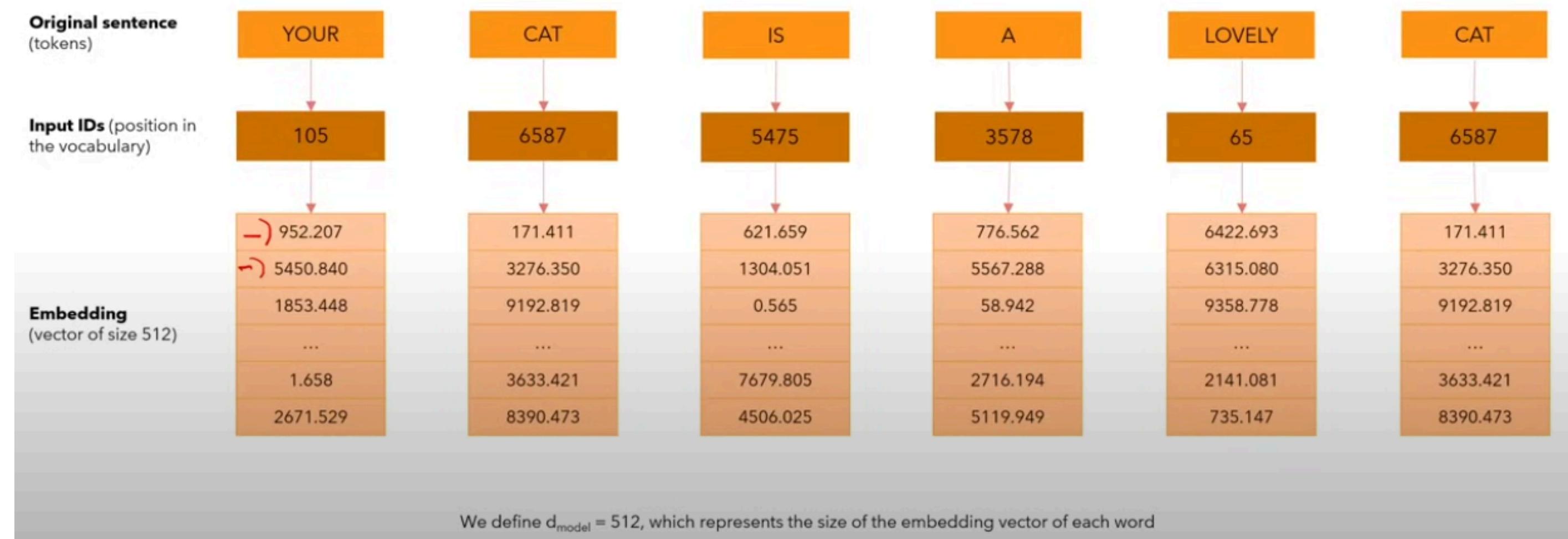


- **Original sentence (tokens):** Câu gốc "YOUR CAT IS A LOVELY CAT" được chia thành các token riêng biệt.
- **Input IDs (position in the vocabulary):** Mỗi token được chuyển đổi thành một số nguyên đại diện cho vị trí của nó trong từ điển của mô hình. Ví dụ, "YOUR" có ID là 105, "CAT" có ID là 6587.
- **Embedding (vector of size 512):** Mỗi ID đầu vào được chuyển đổi thành một vector nhúng có kích thước 512. Đây là biểu diễn số học của mỗi từ trong không gian đa chiều. Chỉ một số giá trị đầu và cuối của vector được hiển thị trong hình.

## 2. Mô hình Transformer

### Encoder

#### Input Embedding



Mô hình có một bảng embedding được đào tạo trước, thường được gọi là Embedding Matrix.

Bảng này có kích thước [vocabulary\_size, embedding\_dim], trong trường hợp này là [vocab\_size, 512].

Mỗi Input ID được sử dụng như một chỉ mục để tra cứu trong bảng embedding.

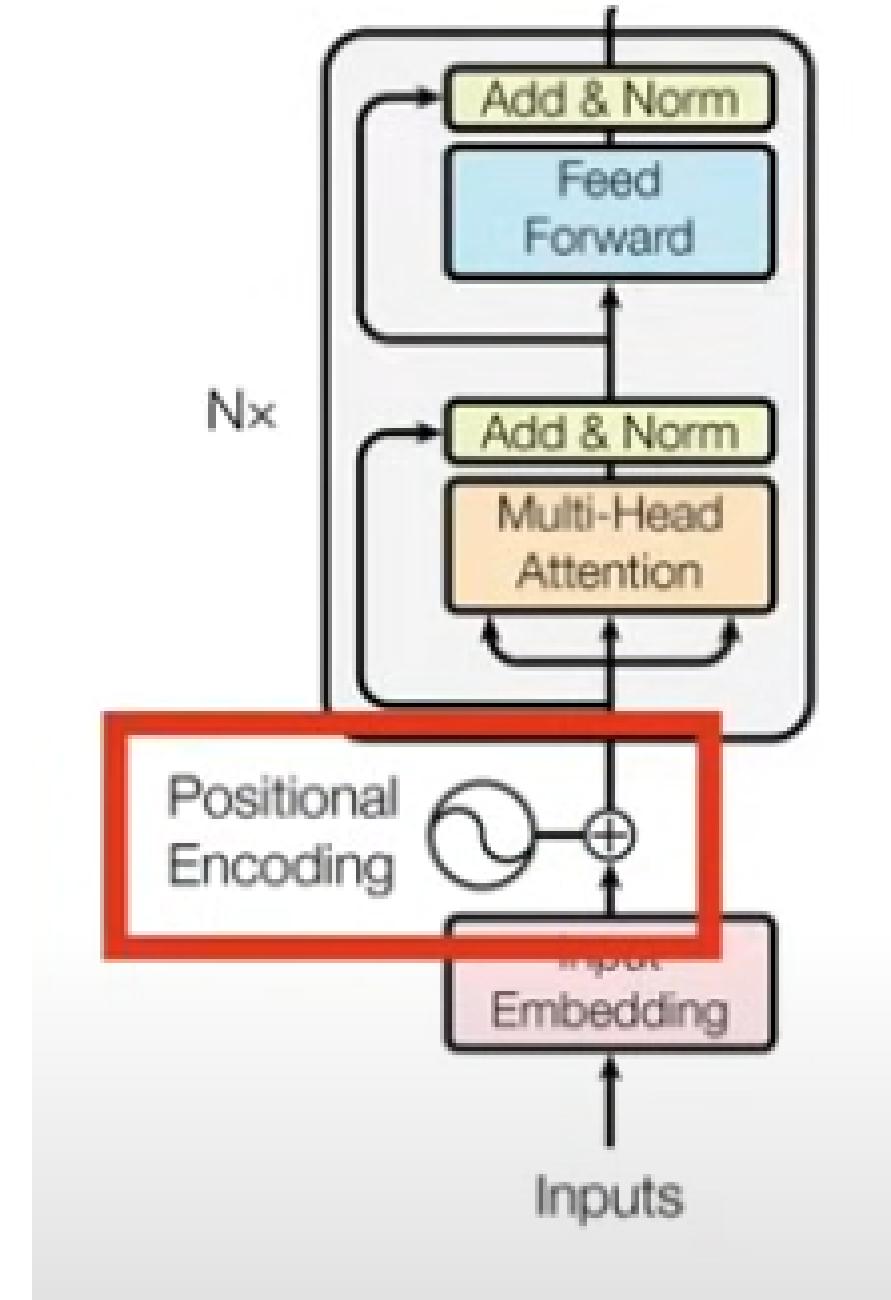
**Ví dụ:** Nếu "CAT" có Input ID là 6587, mô hình sẽ lấy vector thứ 6587 từ bảng embedding.

## 2. Mô hình Transformer

### Encoder

#### Positional Encoding

- Giúp chứa thông tin vị trí của từng từ trong câu
- Mô hình cần phải hiểu mối quan hệ ngữ nghĩa giữa các từ dựa trên vị trí của chúng trong câu.



## 2. Mô hình Transformer

# Encoder

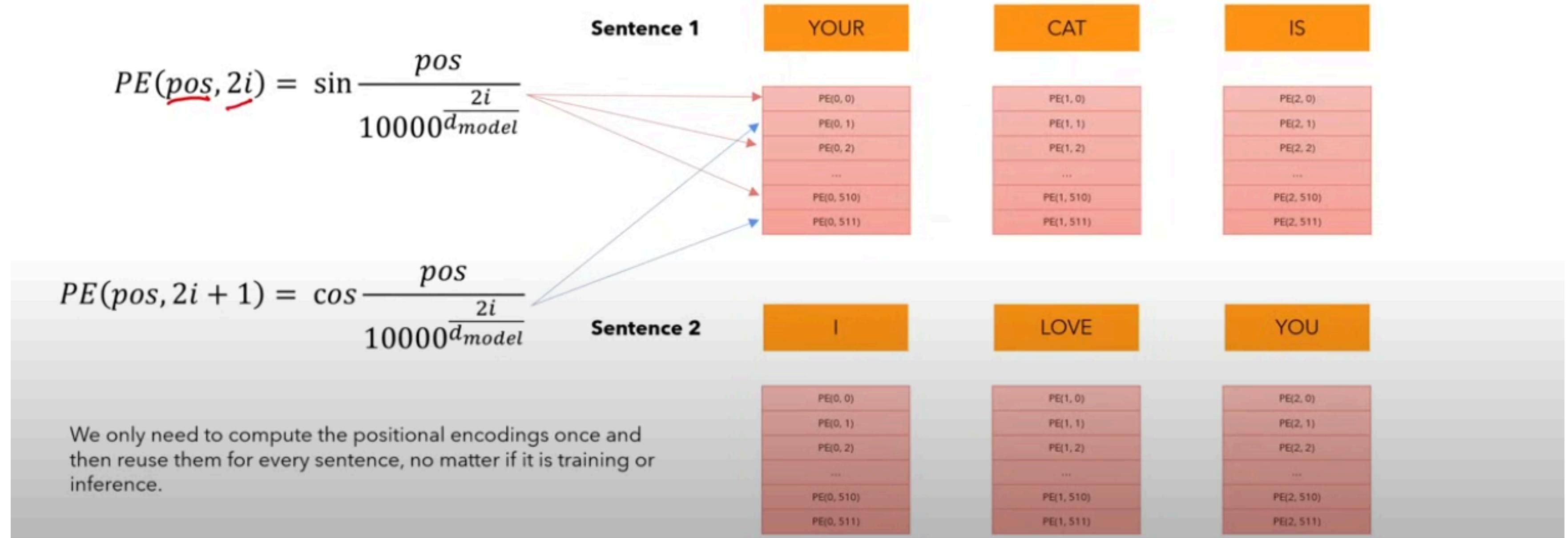
# Positional Encoding



## 2. Mô hình Transformer

### Encoder

#### Positional Encoding

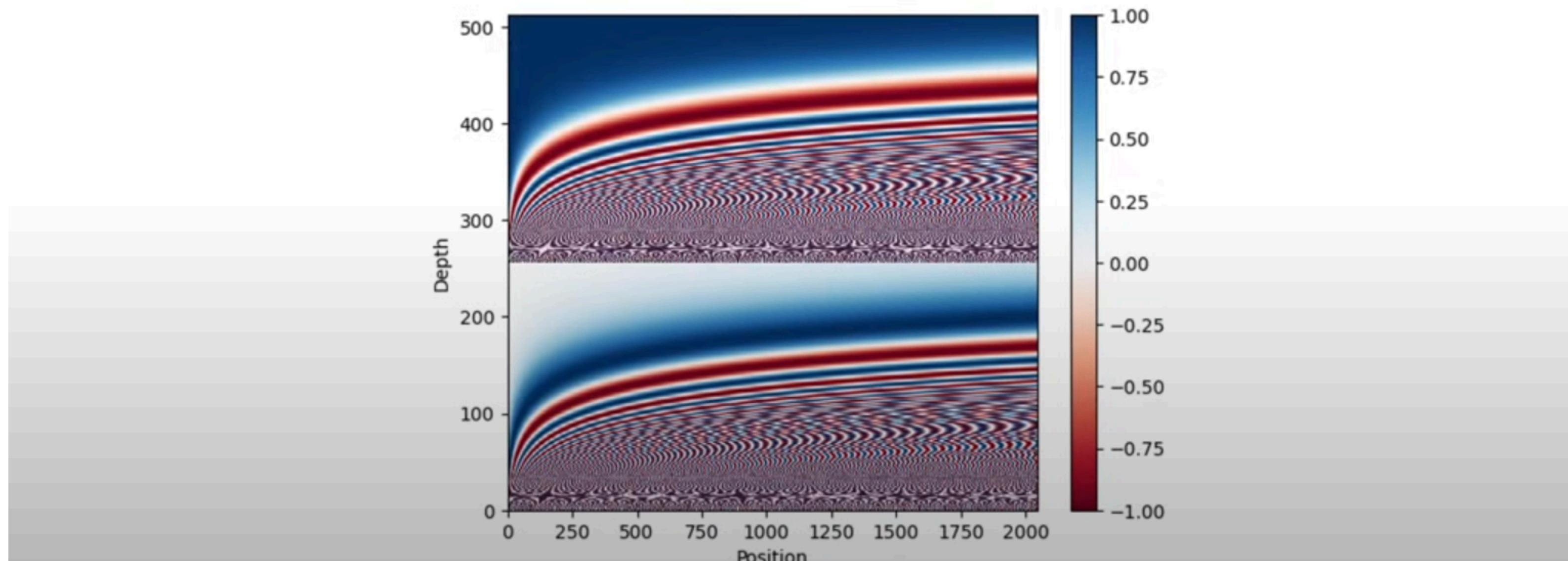


## 2. Mô hình Transformer

### Encoder

#### Positional Encoding

Trigonometric functions like **cos** and **sin** naturally represent a pattern that the model can recognize as continuous, so relative positions are easier to see for the model. By watching the plot of these functions, we can also see a regular pattern, so we can hypothesize that the model will see it too.

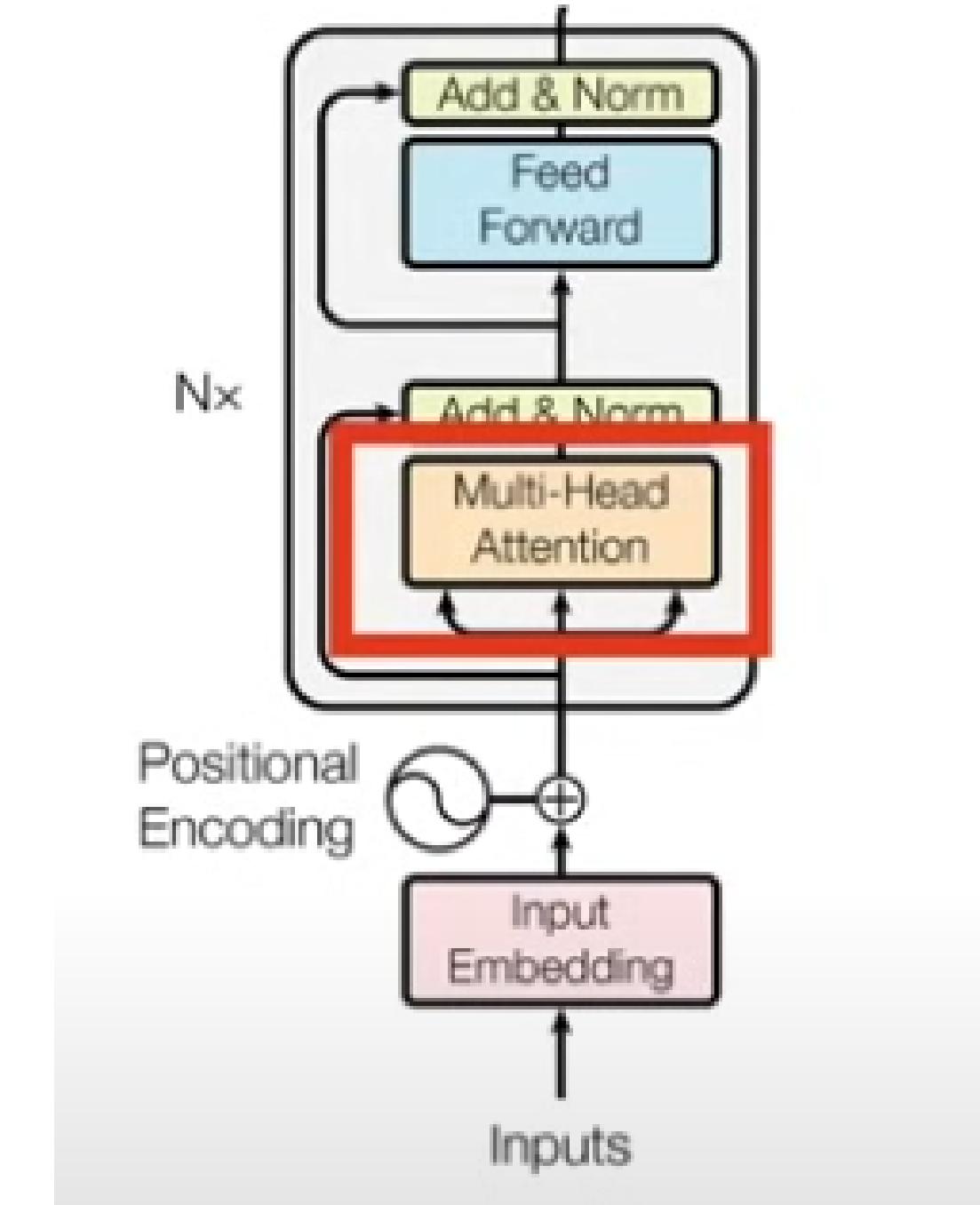


## 2. Mô hình Transformer

### Encoder

#### Multi-Head Attention

Cho phép mô hình tập trung vào nhiều phần khác nhau của chuỗi dữ liệu đồng thời. Mỗi head học từ các khía cạnh khác nhau của thông tin, giúp mô hình nắm bắt tốt hơn các mối quan hệ phức tạp và đa chiều trong dữ liệu, từ đó cải thiện hiệu suất học và biểu diễn.



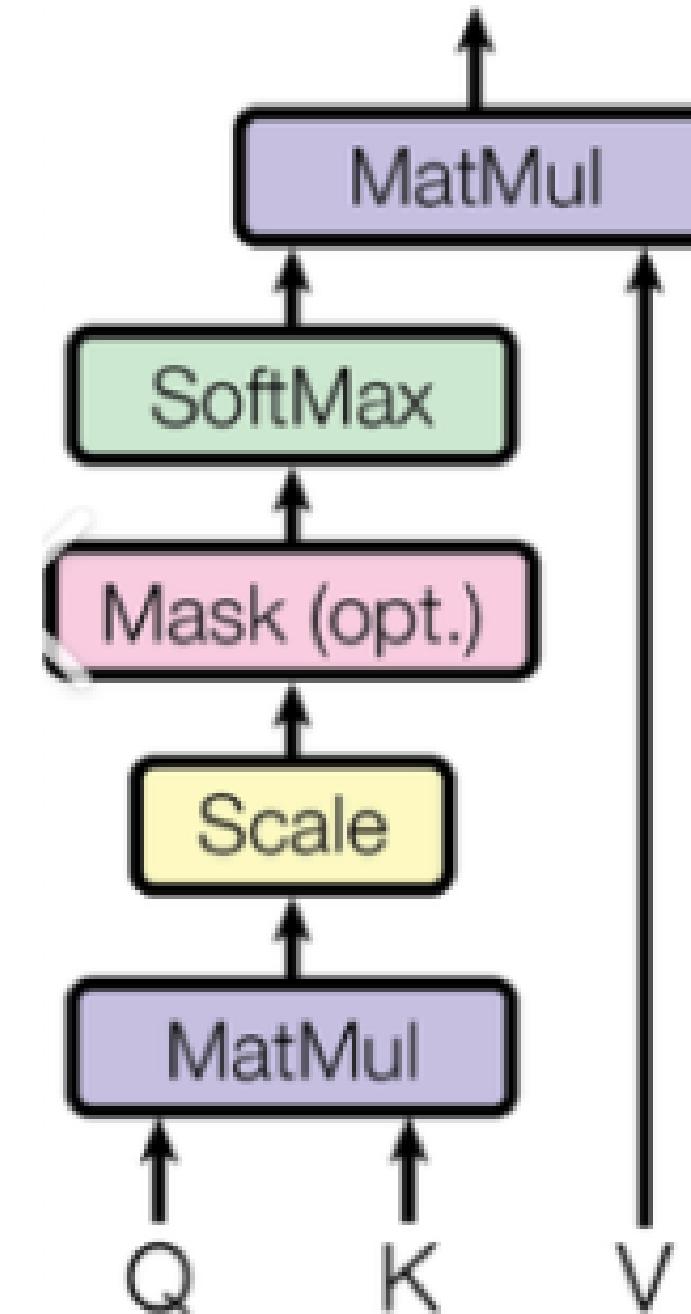
## 2. Mô hình Transformer

### Encoder

#### Self-Attention

Trước khi đi vào Multihead-Attention cần hiểu Self-Attention là gì?

- Self-attention là cơ chế trong mô hình Transformer cho phép mỗi từ trong câu tính toán mối quan hệ với tất cả các từ khác trong cùng câu đó.
- Nó giúp mô hình hiểu ngữ cảnh tổng thể bằng cách tập trung vào các từ quan trọng, bất kể khoảng cách của chúng trong câu.
- Kết quả là mô hình có thể hiểu sâu hơn về ngữ nghĩa và ngữ cảnh của từng từ trong câu.



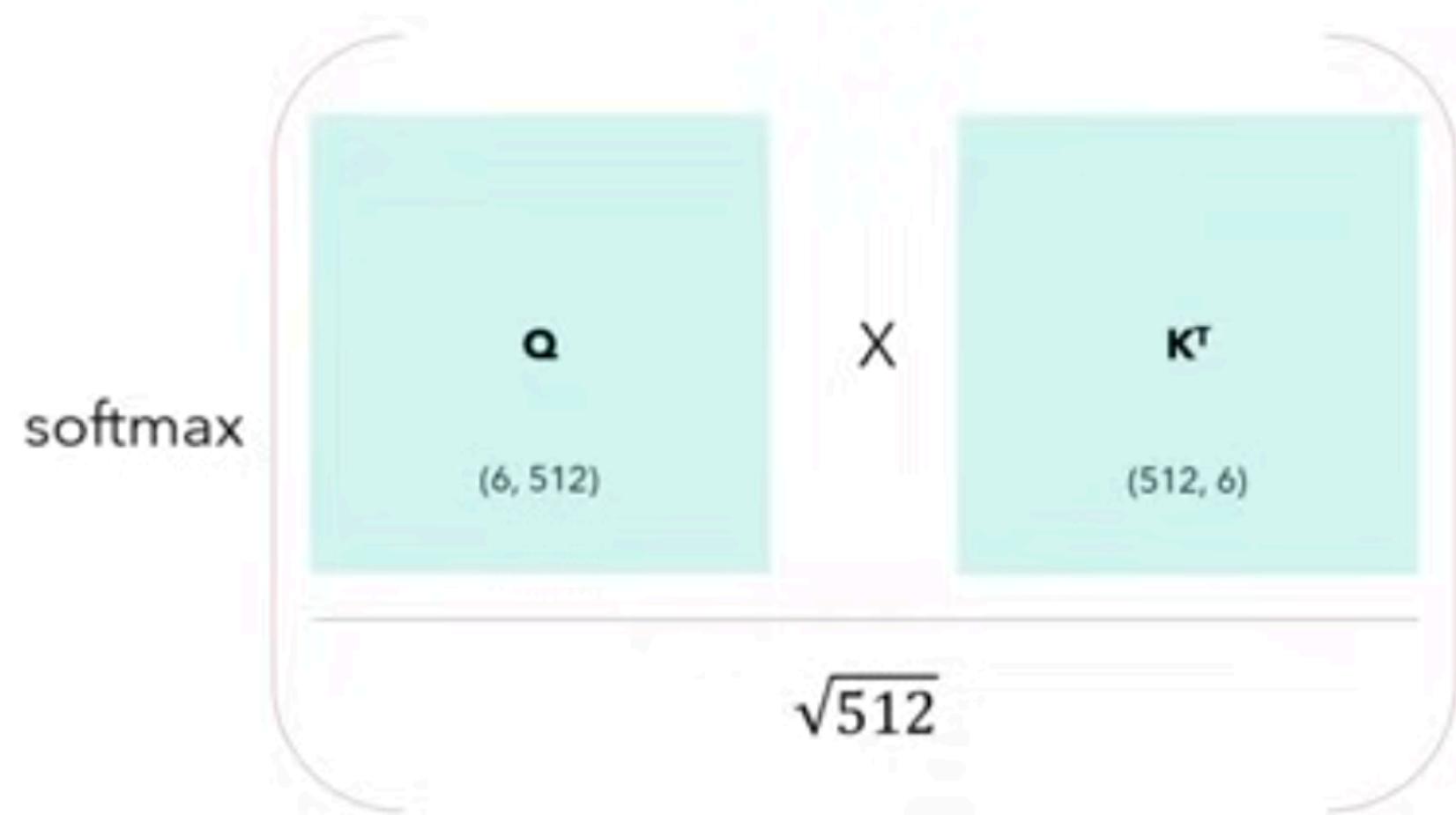
## 2. Mô hình Transformer

### Encoder Self-Attention

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length  $\text{seq} = 6$  and  $\mathbf{d}_{\text{model}} = \mathbf{d}_k = 512$ .

The matrices  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$  are just the input sentence.



	YOUR	CAT	IS	A	LOVELY	CAT	I
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

\* all values are random.

\* for simplicity I considered only one head, which makes  $\mathbf{d}_{\text{model}} = \mathbf{d}_k$ .

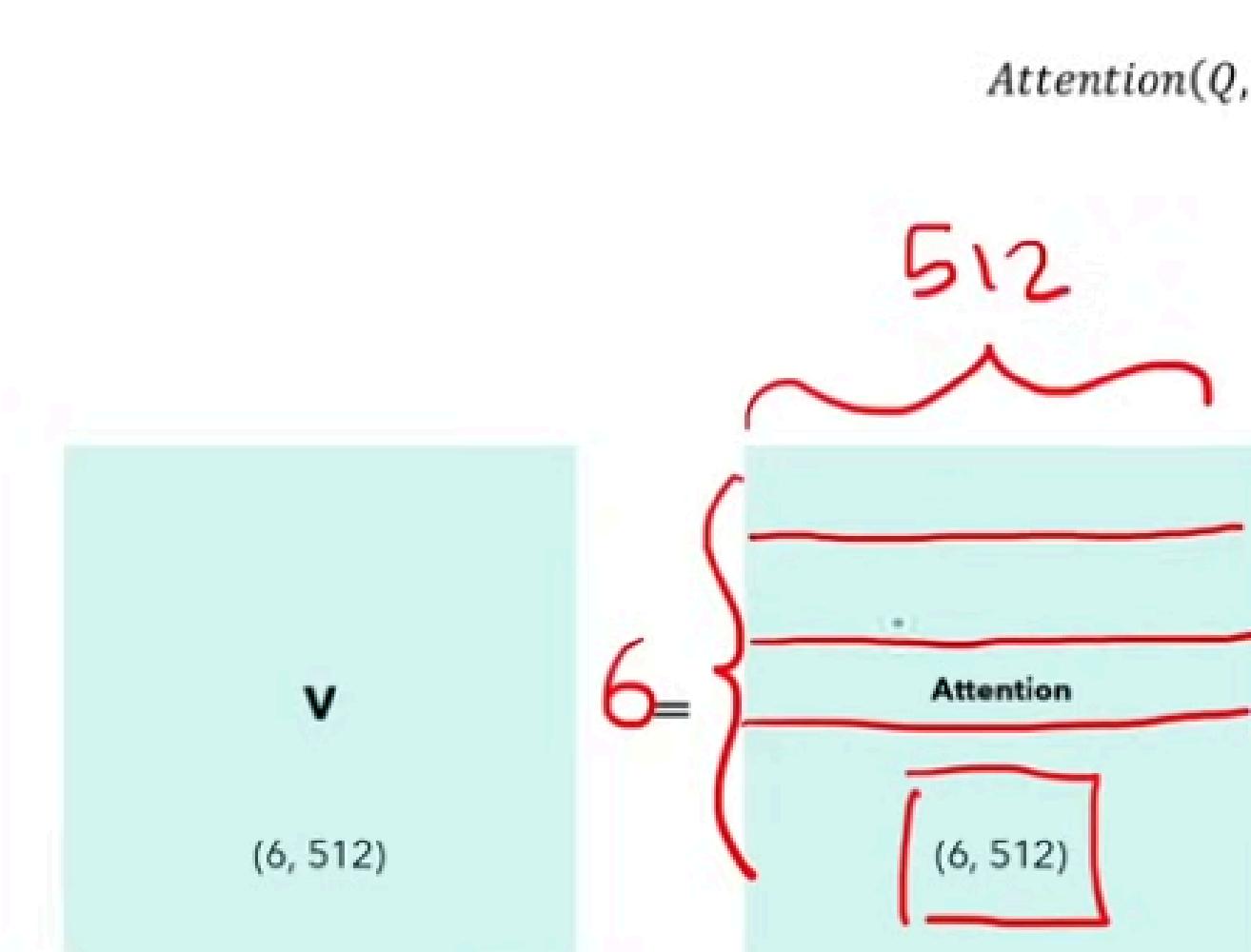
(6, 6)

## 2. Mô hình Transformer

### Encoder Self-Attention

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)



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

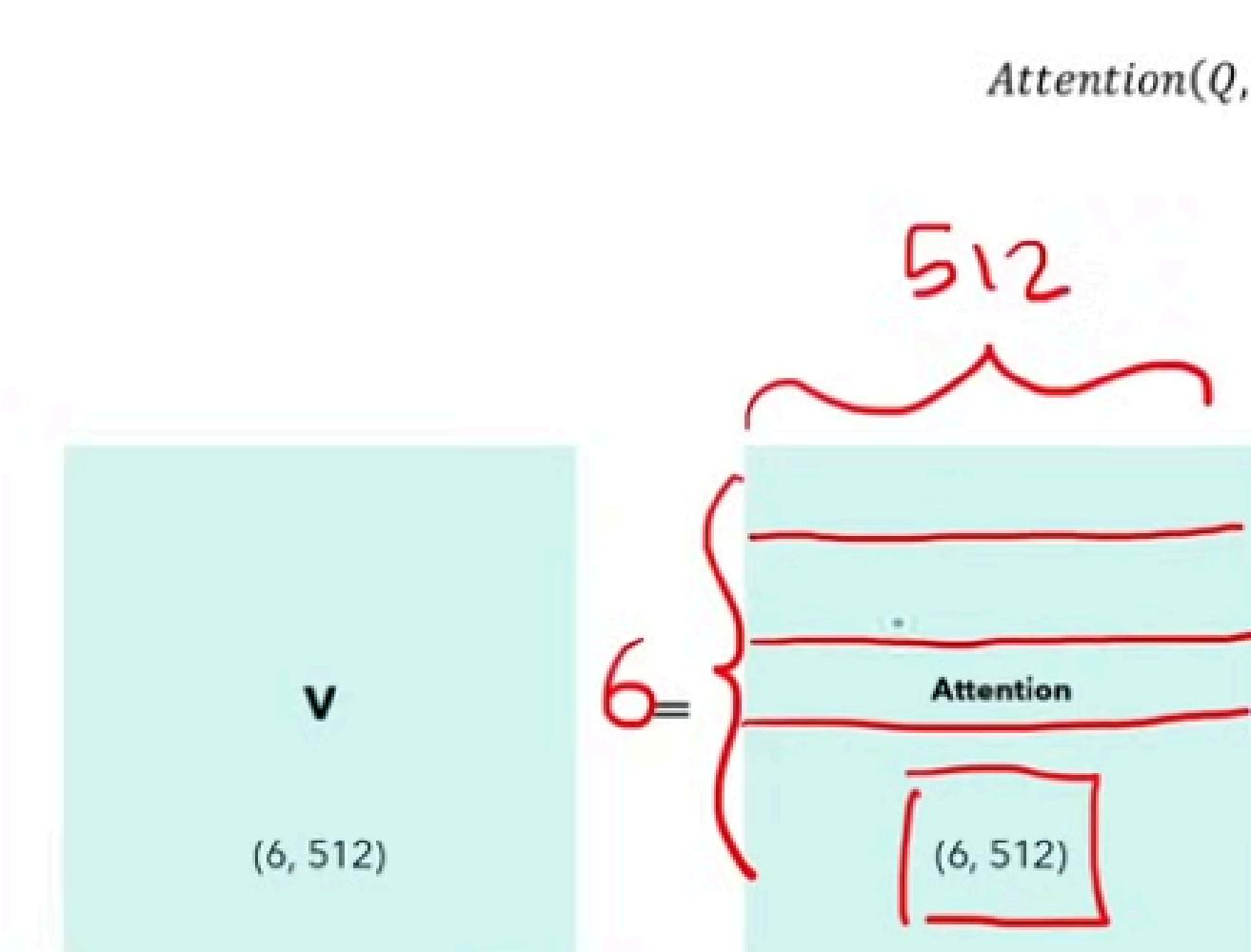
Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

## 2. Mô hình Transformer

### Encoder Self-Attention

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)



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

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

## 2. Mô hình Transformer

### Encoder

#### Multi-Head Attention

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

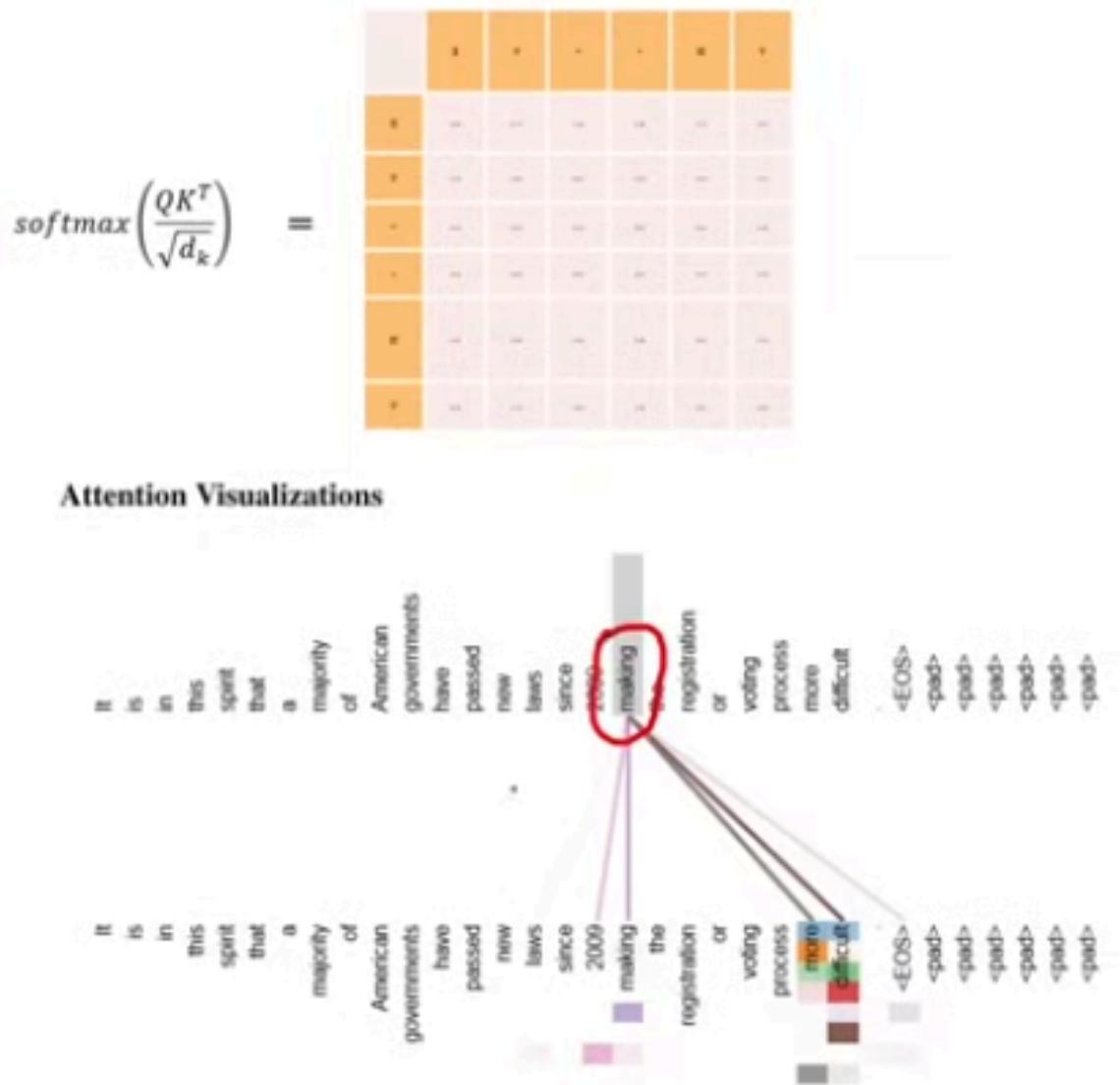
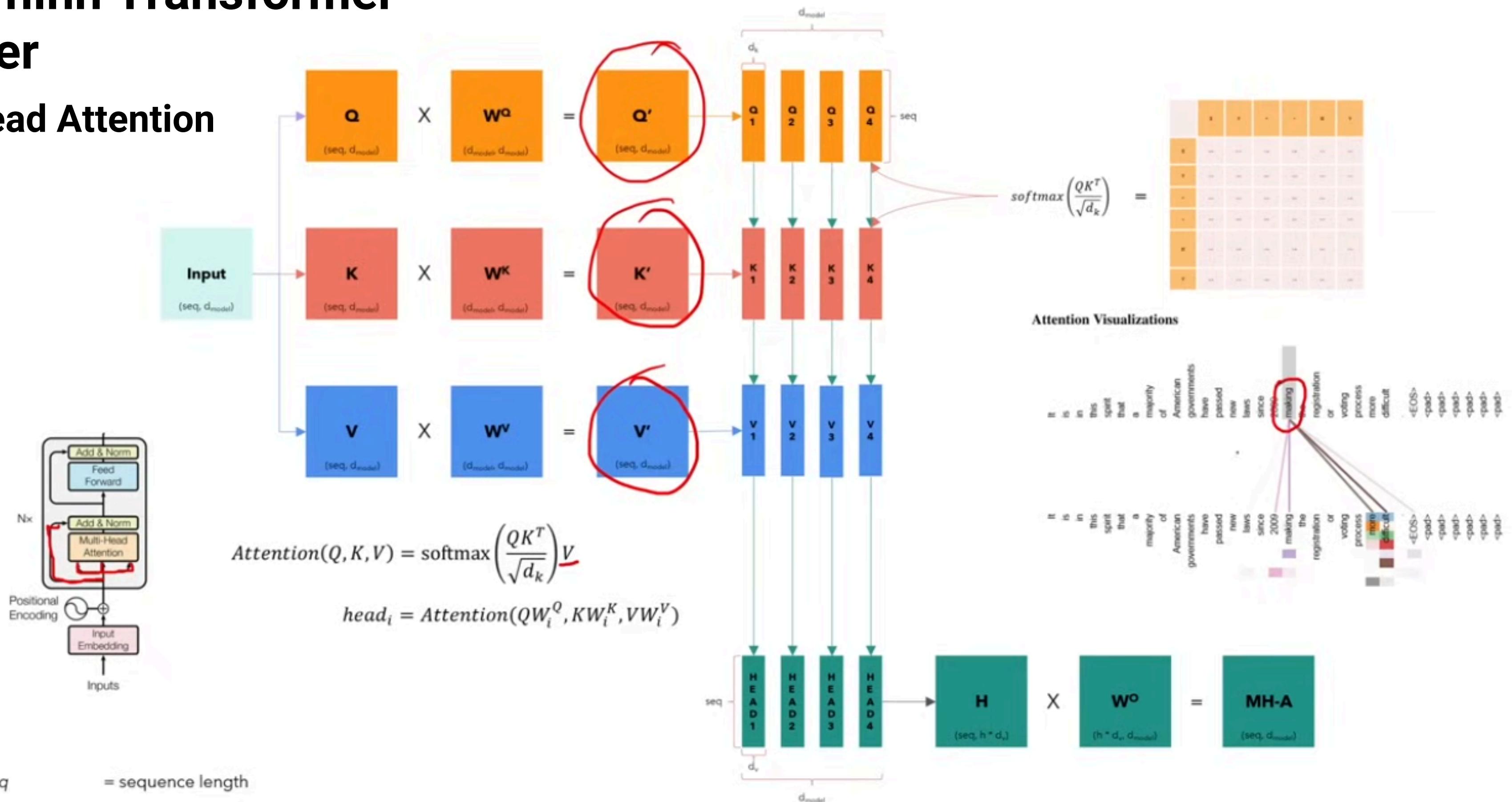
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \dots \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- Thay vì chỉ sử dụng một cơ chế attention duy nhất, multi-head attention tạo ra nhiều "head" attention, mỗi "head" có một cách nhìn khác nhau về mối quan hệ giữa các từ trong câu.
- Mỗi "head" thực hiện một phép tính self-attention độc lập. Sau đó, các kết quả của từng "head" được kết hợp lại và chuyển qua một lớp tuyến tính để tổng hợp thành một đầu ra cuối cùng.

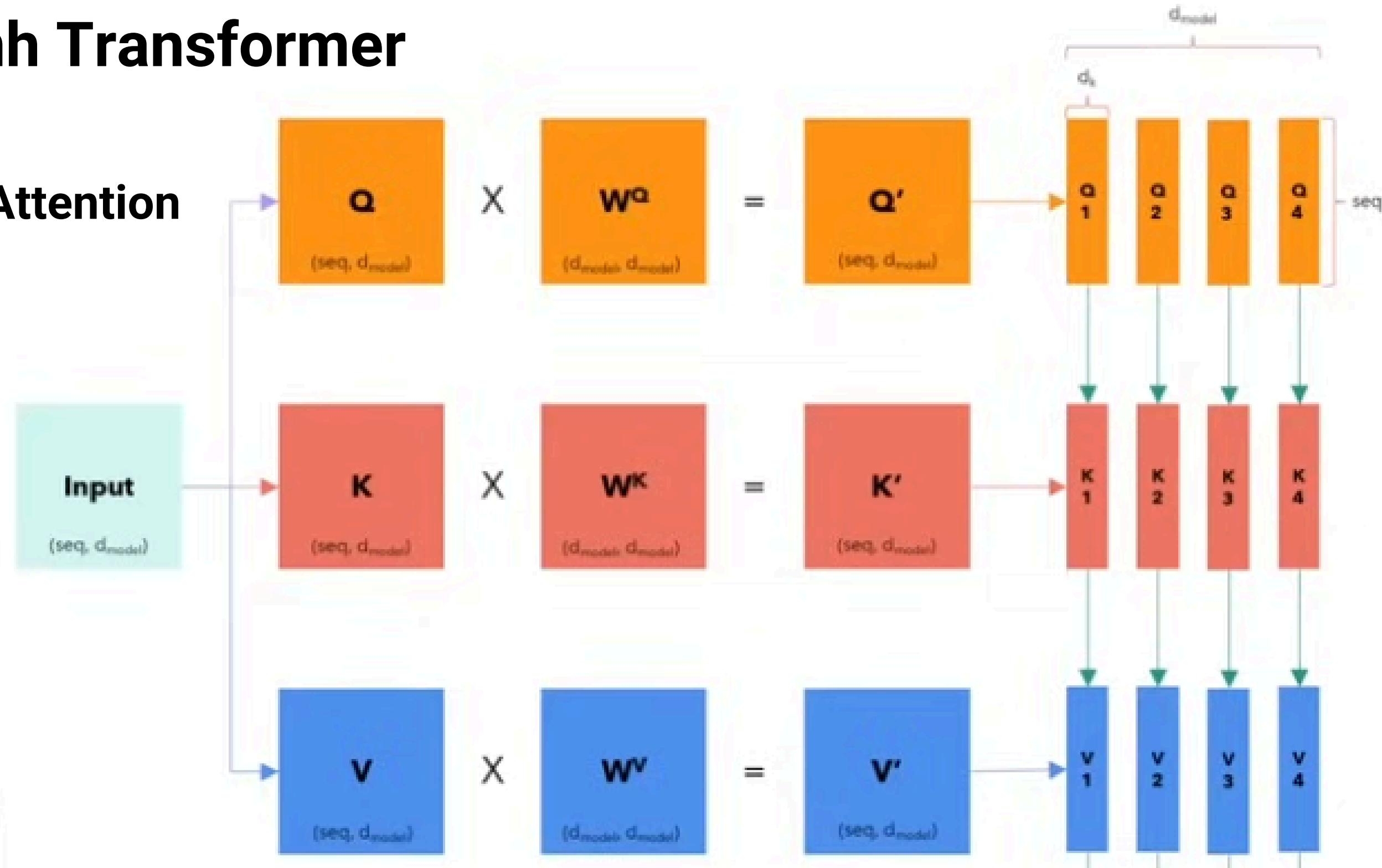
# 2. Mô hình Transformer Encoder

## Multi-Head Attention



## 2. Mô hình Transformer Encoder

### Multi-Head Attention



$seq$

= sequence length

$d_{model}$

= size of the embedding vector

$h$

= number of heads

$d_h = d_{model} / h$

- Việc sử dụng nhiều attention head với kích thước nhỏ hơn cho mỗi đầu chú ý cho phép mô hình tập trung vào nhiều khía cạnh khác nhau của dữ liệu đồng thời. Mặc dù tổng chi phí tính toán không thay đổi so với việc sử dụng một attention head duy nhất với kích thước đầy đủ, mô hình vẫn có khả năng học và nắm bắt thông tin từ nhiều khía cạnh diễn khía cạnh khác nhau, nhờ vào sự phân chia và xử lý đồng thời này.

## 2. Mô hình Transformer

### Encoder

#### Tại sao sử dụng các ma trận Query, Key, Value



The key/value/query formulation of attention is from the paper [Attention Is All You Need](#).



280 How should one understand the queries, keys, and values

The key/value/query concept is analogous to retrieval systems. For example, when you search for videos on Youtube, the search engine will map your **query** (text in the search bar) against a set of **keys** (video title, description, etc.) associated with candidate videos in their database, then present you the best matched videos (**values**).

The attention operation can be thought of as a retrieval process as well.

As mentioned in the paper you referenced ([Neural Machine Translation by Jointly Learning to Align and Translate](#)), attention by definition is just a weighted average of values,

$$c = \sum_j \alpha_j h_j$$

where  $\sum \alpha_j = 1$ .

<https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms>

## 2. Mô hình Transformer

### Encoder

#### Add & Norm

Ổn định và tăng tốc quá trình huấn luyện bằng cách chuẩn hóa đầu vào của mỗi lớp (layer) sao cho chúng có cùng phân phối. Giúp giảm thiểu sự thay đổi phân phối của dữ liệu trong quá trình huấn luyện, làm cho mô hình học hiệu quả hơn và tránh hiện tượng gradient vanishing hoặc exploding.

Batch of 3 items

ITEM 1

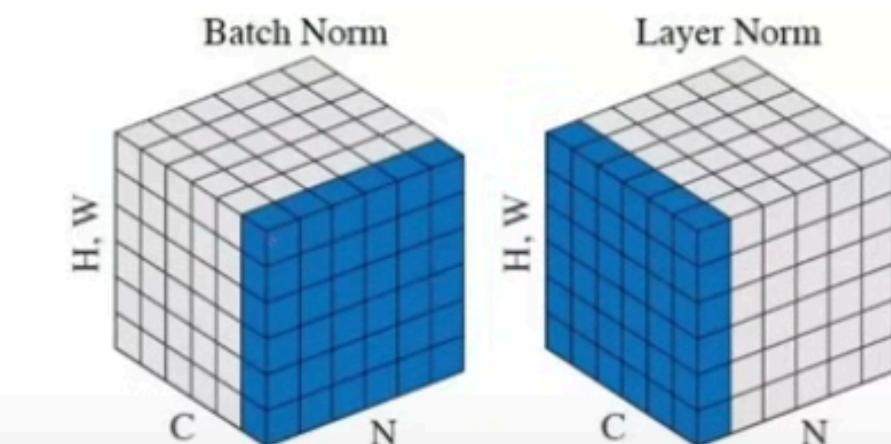
50.147
3314.825
...
...
8463.361
8.021

ITEM 2

1242.223
688.123
...
...
434.944
149.442

ITEM 3

9.370
4606.674
...
...
944.705
21189.444



$$\mu_1$$

$$\sigma_1^2$$

$$\mu_2$$

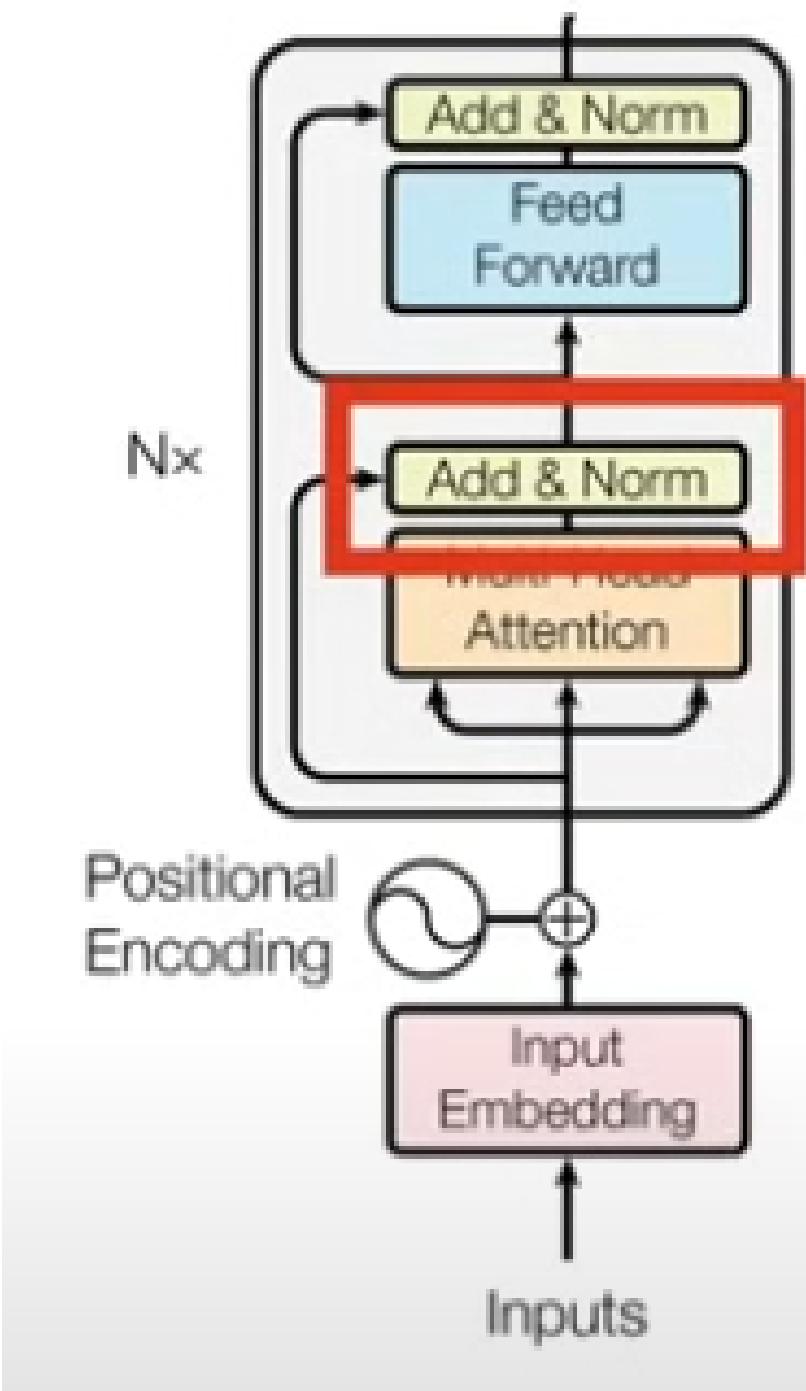
$$\sigma_2^2$$

$$\mu_3$$

$$\sigma_3^2$$

$$\hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

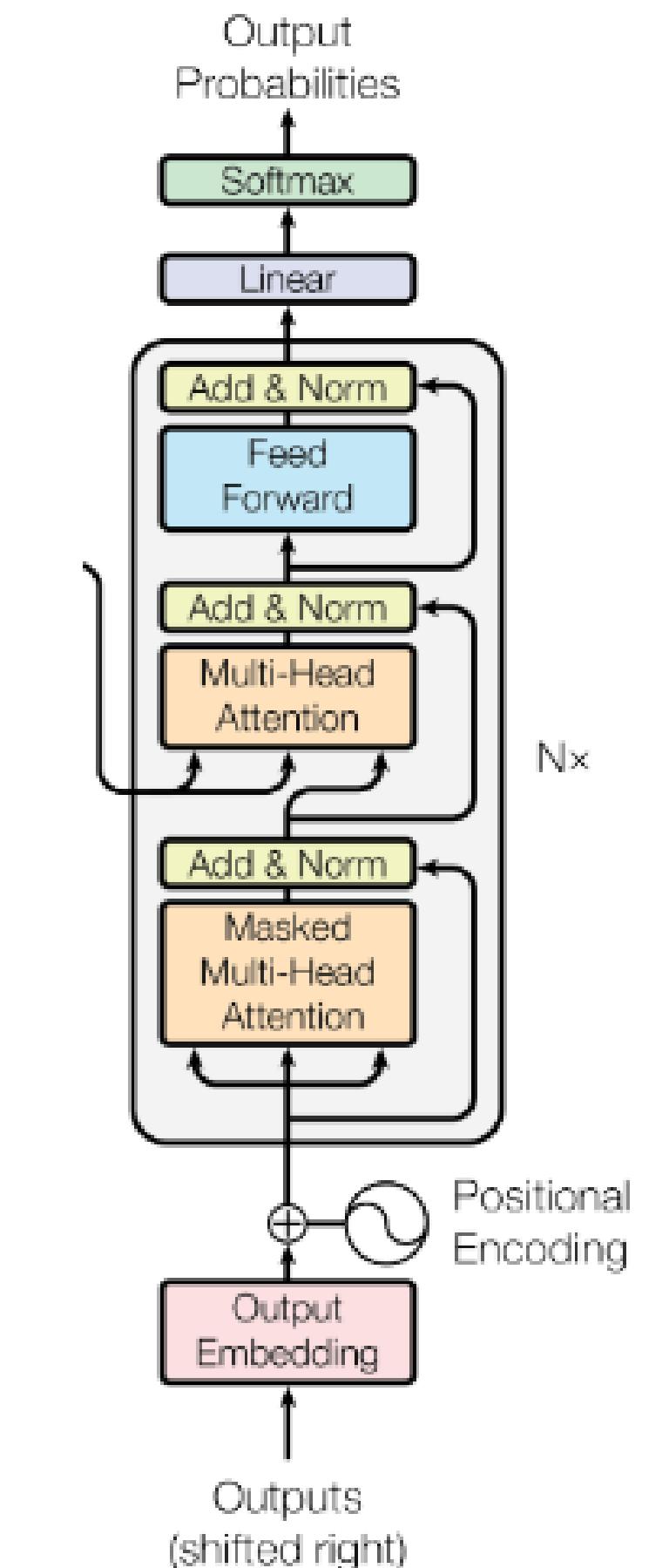
We also introduce two parameters, usually called **gamma** (multiplicative) and **beta** (additive) that introduce some fluctuations in the data, because maybe having all values between 0 and 1 may be too restrictive for the network. The network will learn to tune these two parameters to introduce fluctuations when necessary.



## 2. Mô hình Transformer

### Decoder

Tạo ra đầu ra dựa trên các thông tin đã được mã hóa bởi khối Encoder và thông tin đã sinh ra trước đó.

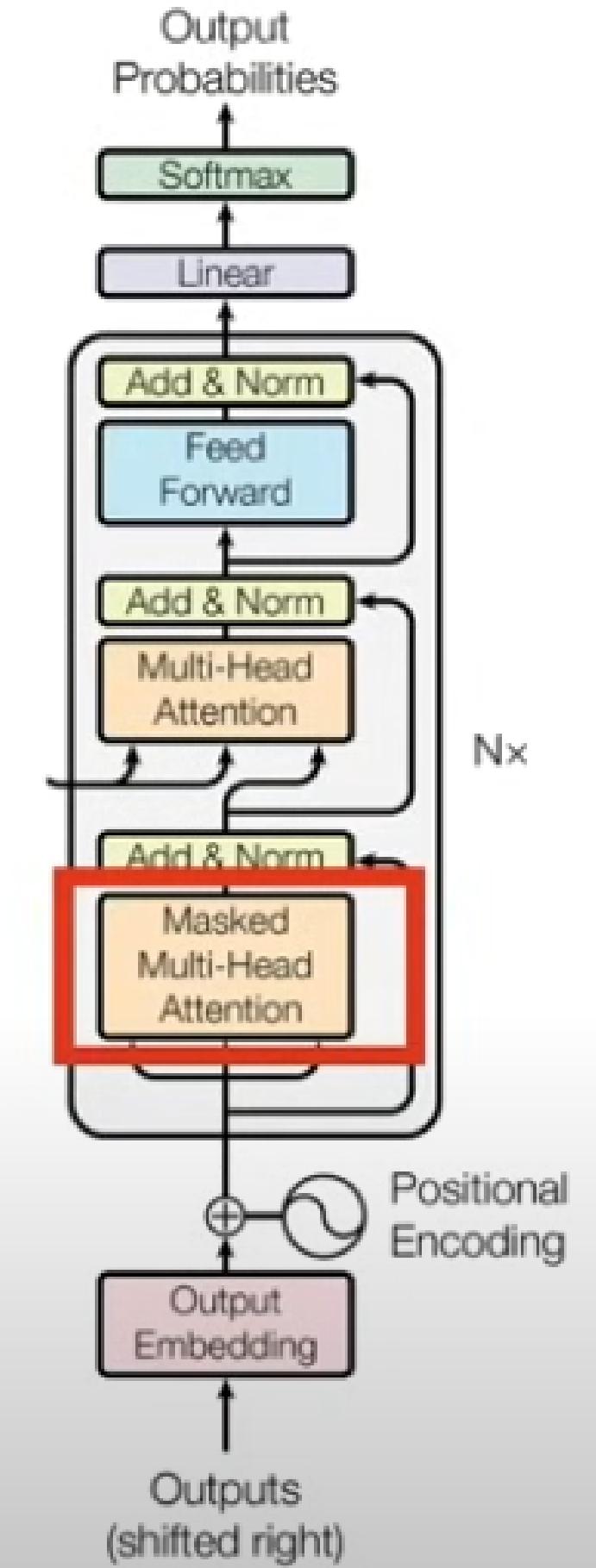


## 2. Mô hình Transformer

### Decoder

#### Masked Multi-head Attention

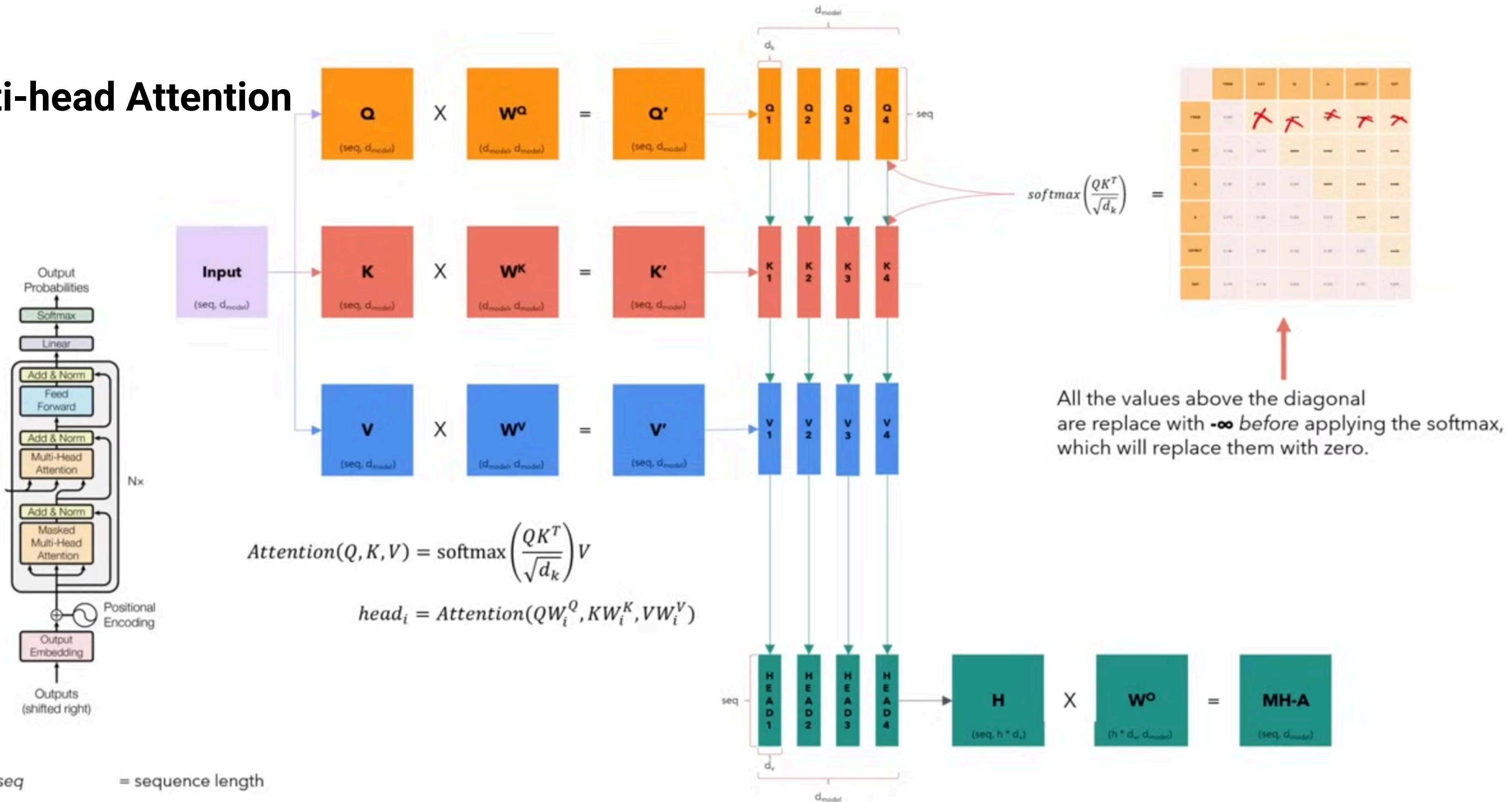
- Ngăn mô hình nhìn trước thông tin tương lai khi sinh đầu ra.  
Cụ thể, nó tạo ra một "mask" để che đi các token phía sau (chưa sinh ra) trong quá trình tính attention.
- Đảm bảo rằng, tại mỗi bước, mô hình chỉ có thể sử dụng thông tin từ các token đã có trước đó, giúp quá trình sinh chuỗi (ví dụ, dịch ngôn ngữ) diễn ra theo trình tự và hợp lý.



# 2. Mô hình Transformer

## Decoder

### Masked Multi-head Attention



$seq$  = sequence length

$d_{model}$  = size of the embedding vector

$h$  = number of heads

$d_k - d_v$  =  $d_{model} / h$

$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \dots \text{head}_h)W^O$

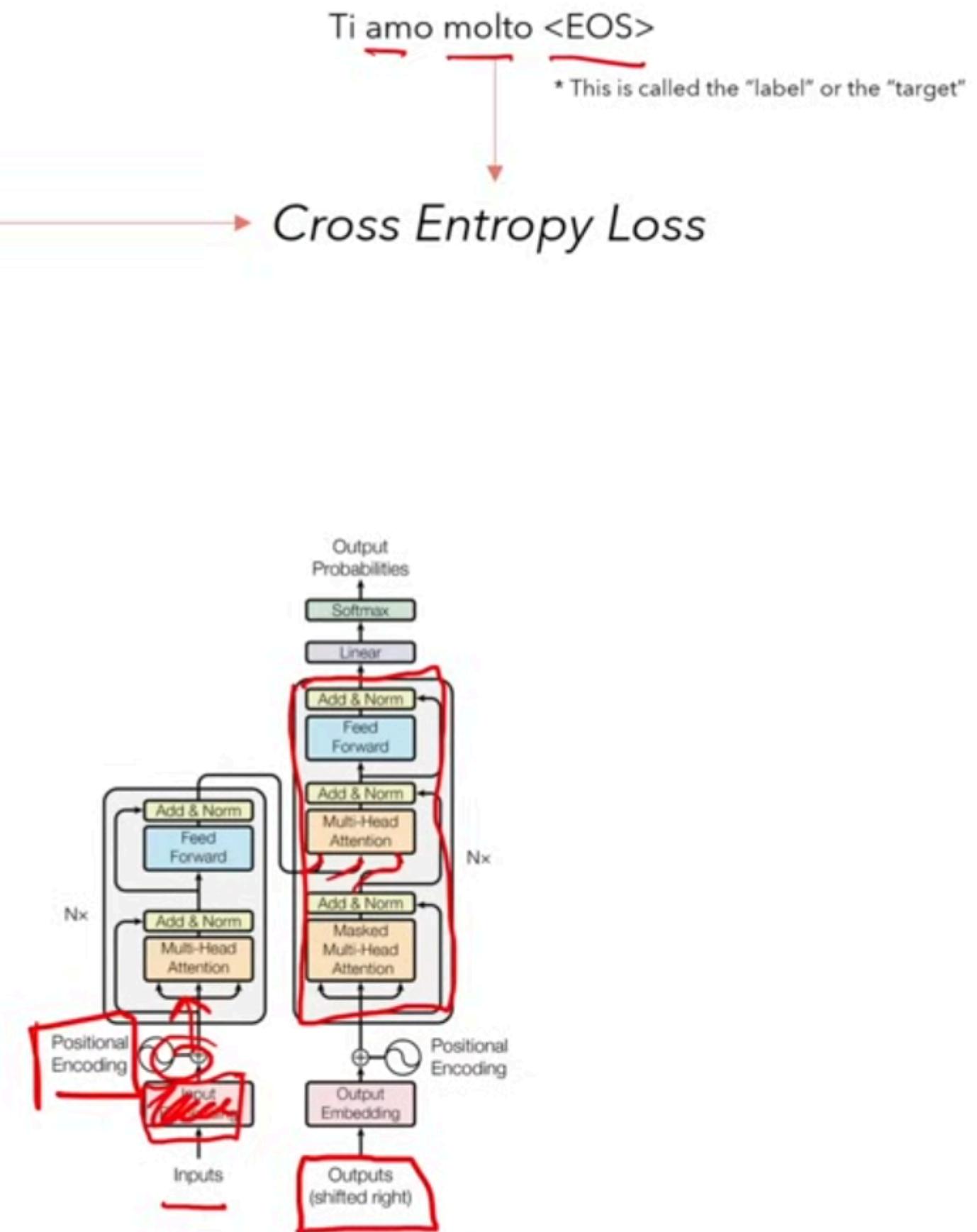
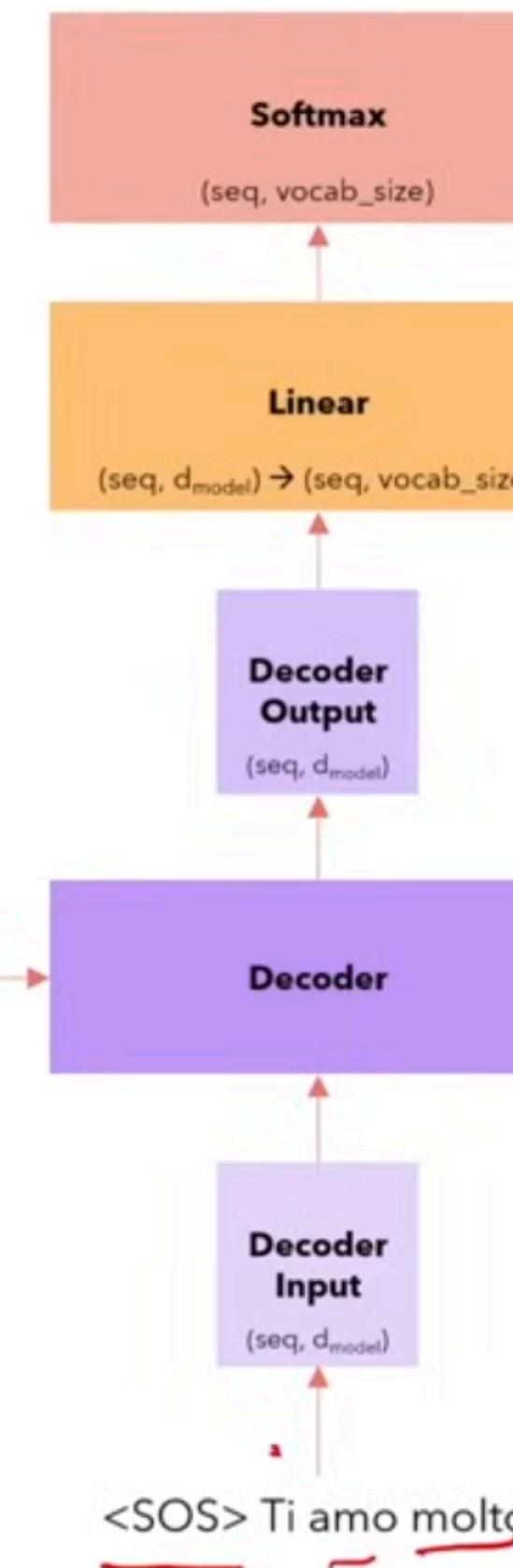
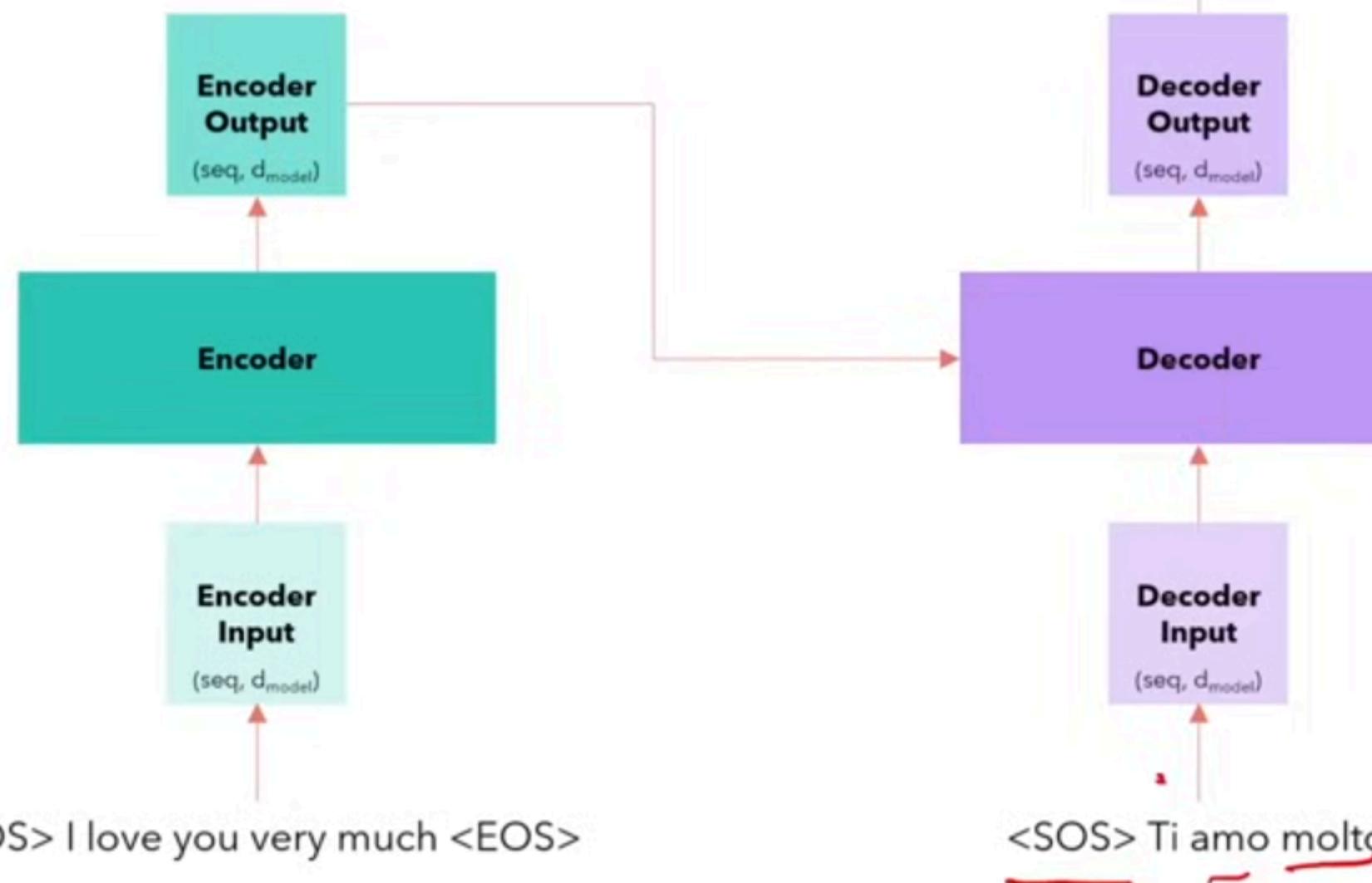
### 3. Quá trình Training

## Training

Time Step = 1

**It all happens in one time step!**

The encoder outputs, for each word a vector that not only captures its meaning (the embedding) or the position, but also its interaction with other words by means of the multi-head attention.

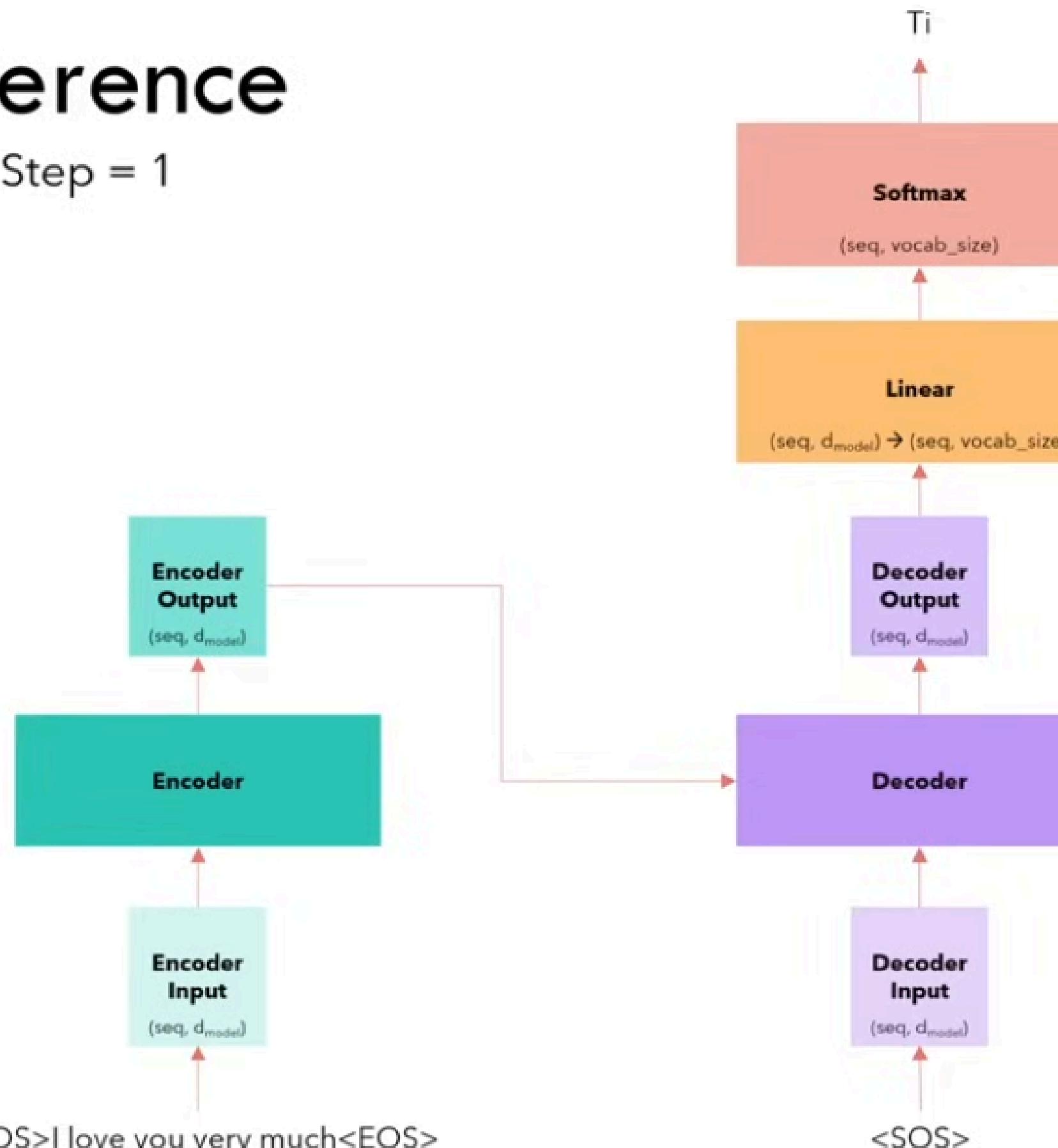


We prepend the <SOS> token at the beginning. That's why the paper says that the decoder input is shifted right.

# 4. Quá trình Inference

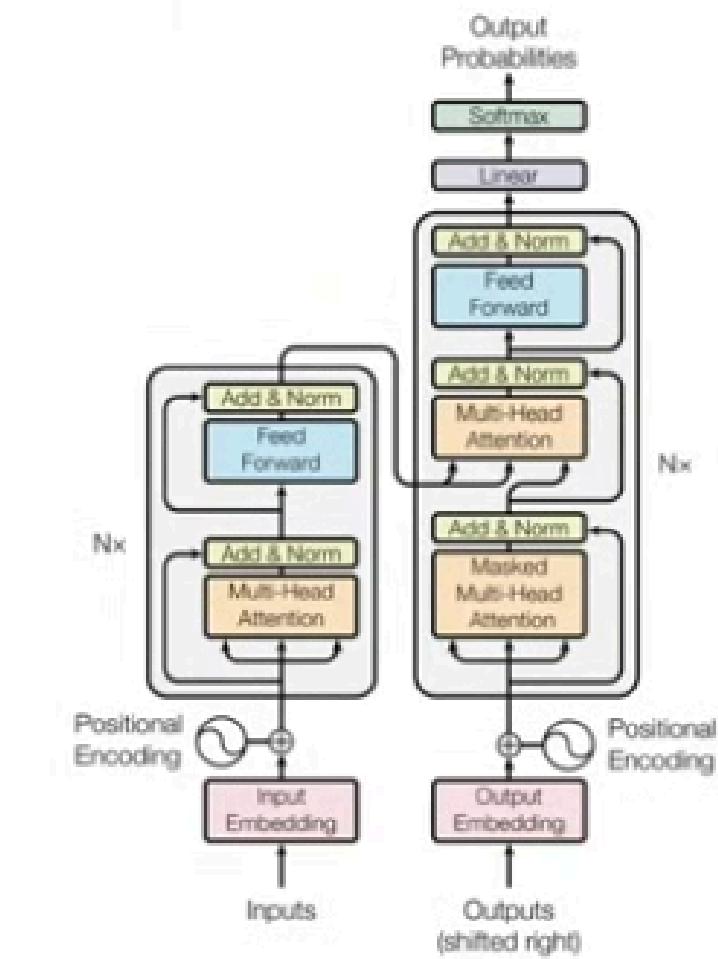
## Inference

Time Step = 1



We select a token from the vocabulary corresponding to the position of the token with the maximum value.

The output of the last layer is commonly known as **logits**



\* Both sequences will have same length thanks to padding

# 4. Quá trình Inference

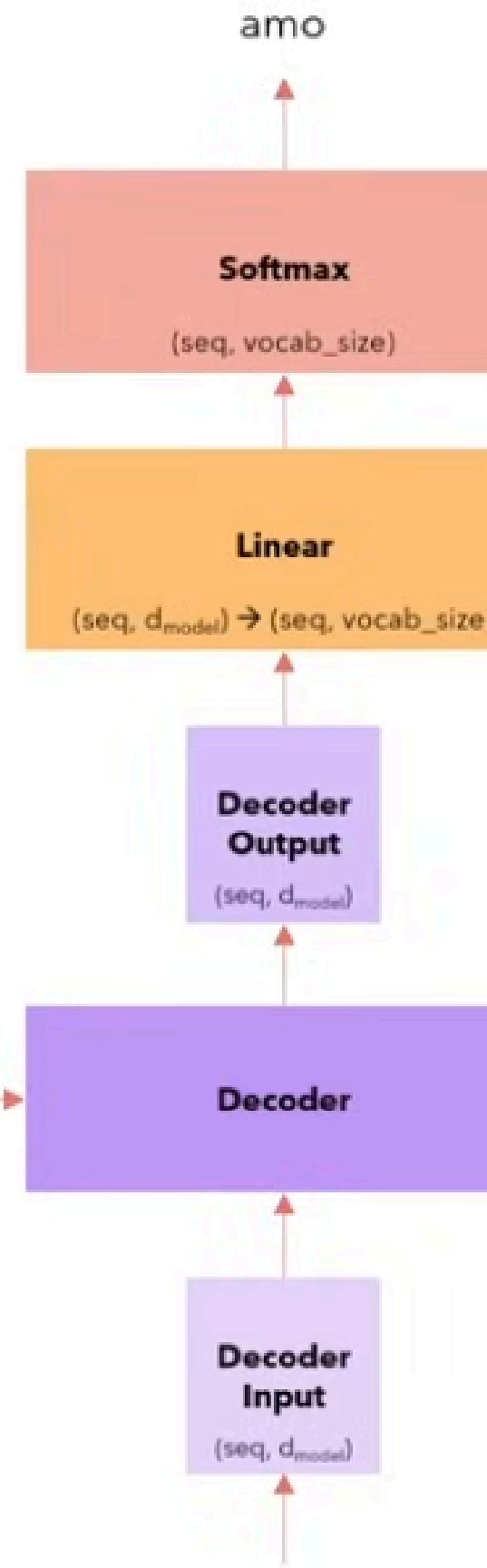
## Inference

Time Step = 2

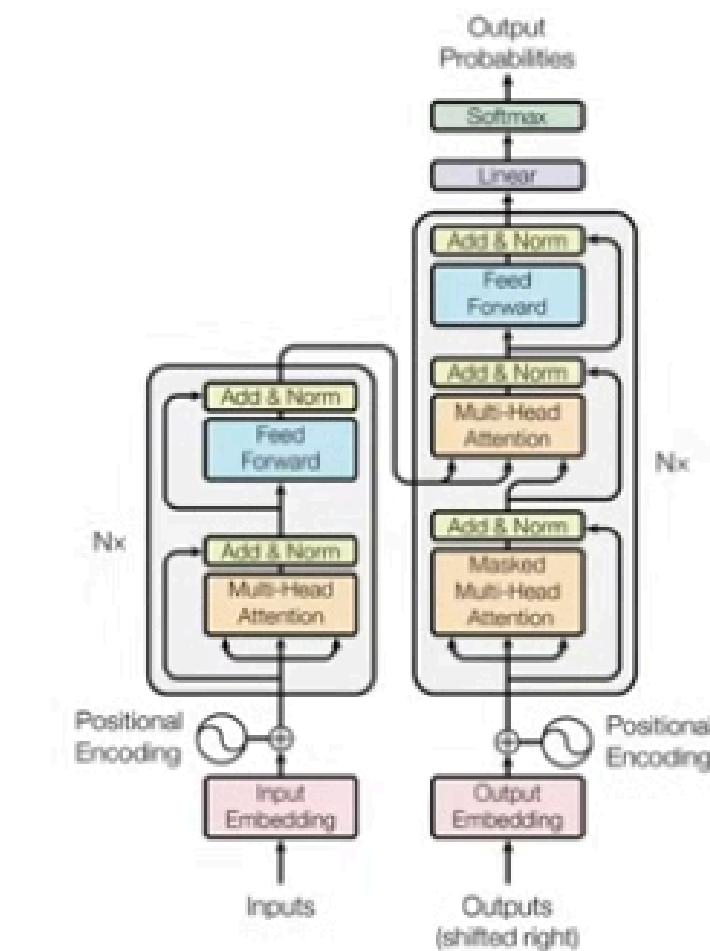
Use the encoder output from the first time step

<SOS>I love you very much<EOS>

<SOS> ti



Since decoder input now contains **two** tokens, we select the softmax corresponding to the second token.



Append the previously output word to the decoder input

# 4. Quá trình Inference

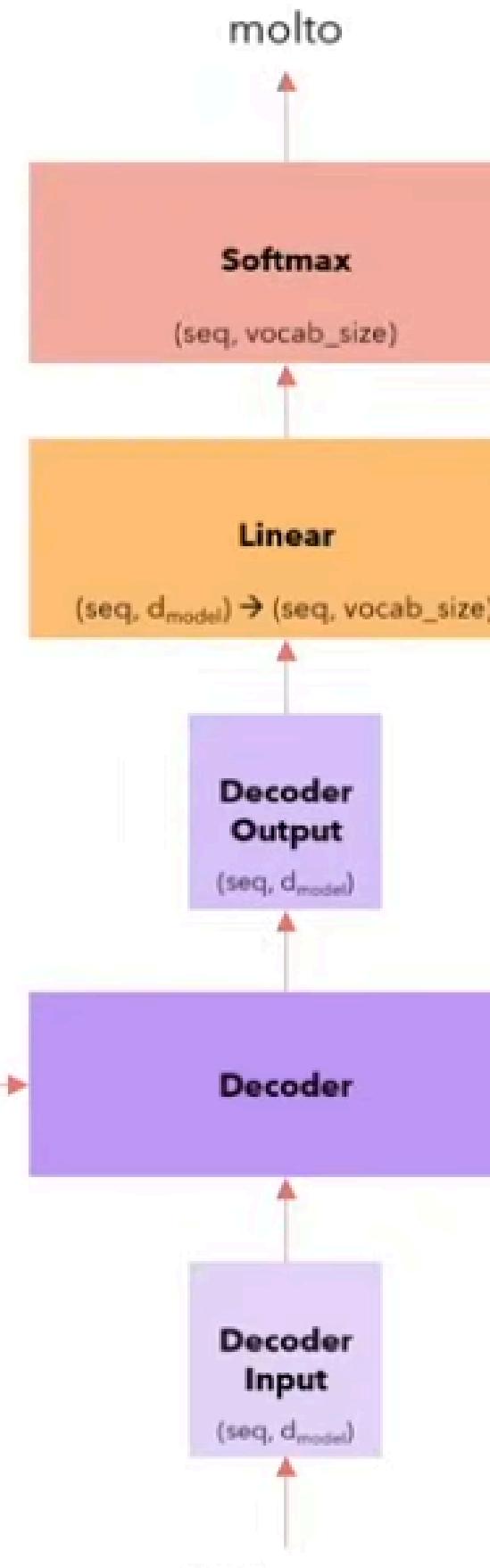
## Inference

Time Step = 3

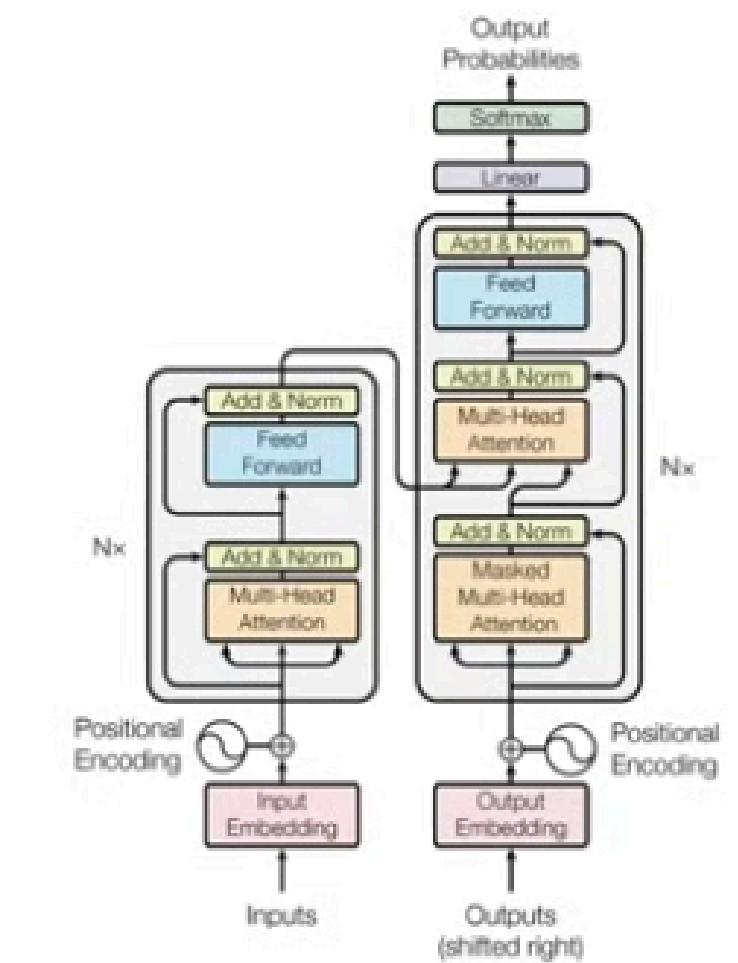
Use the encoder output from the first time step

<SOS>I love you very much<EOS>

<SOS> ti amo



Since decoder input now contains **three** tokens, we select the softmax corresponding to the third token.



Append the previously output word to the decoder input

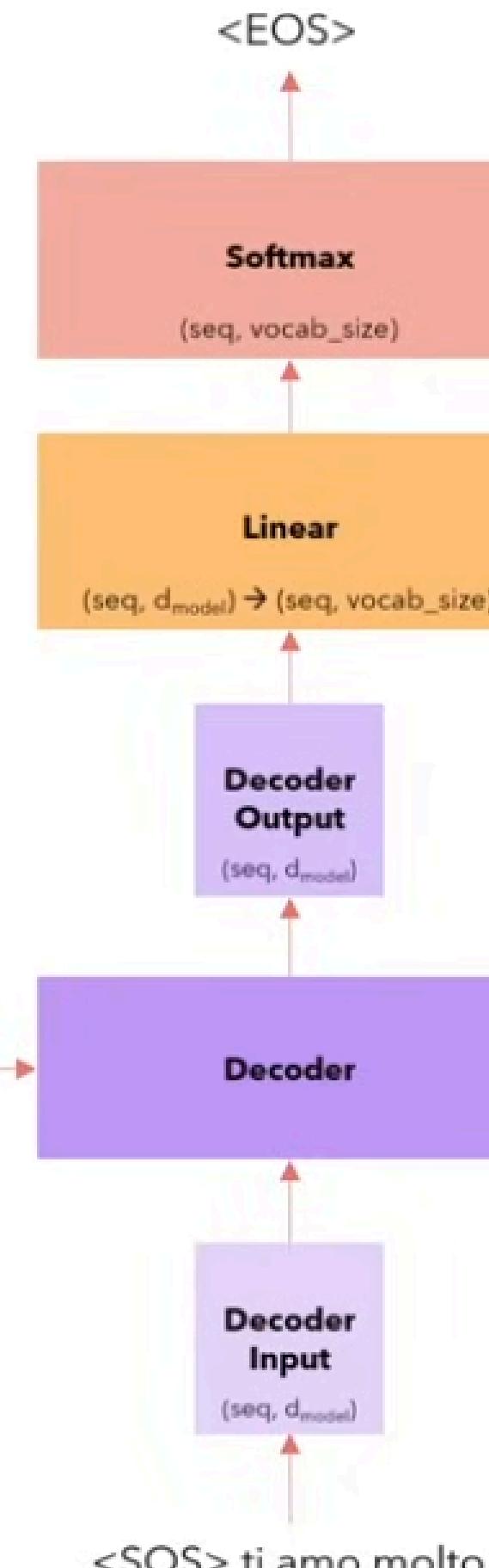
# 4. Quá trình Inference

## Inference

Time Step = 4

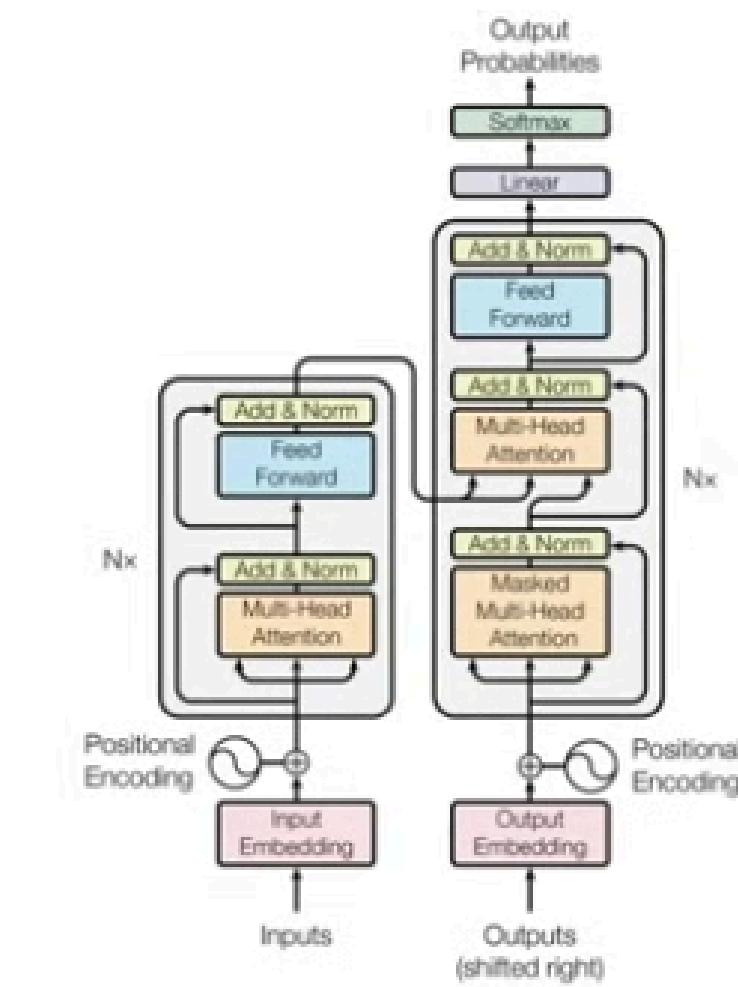
Use the encoder output from the first time step

<SOS>I love you very much<EOS>



Since decoder input now contains **four** tokens, we select the softmax corresponding to the fourth token.

<SOS> ti amo molto



Append the previously output word to the decoder input

## 4. Các chiến lược Inference

### Inference strategy

- We selected, at every step, the word with the maximum softmax value. This strategy is called **greedy** and usually does not perform very well.
- A better strategy is to select at each step the top  $B$  words and evaluate all the possible next words for each of them and at each step, keeping the top  $B$  most probable sequences. This is the **Beam Search** strategy and generally performs better.

# Tham khảo

Paper: <https://arxiv.org/pdf/1706.03762>

Giải thích tại sao dùng Q, K, V <https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms>

Video code chi tiết <https://youtu.be/lSNdQcPhsts?si=WrO9IL98mNWKhcvd>

Video giải thích chi tiết [https://youtu.be/bCz4OMemCcA?si=8oQz-Jyc\\_RmUStGI](https://youtu.be/bCz4OMemCcA?si=8oQz-Jyc_RmUStGI)