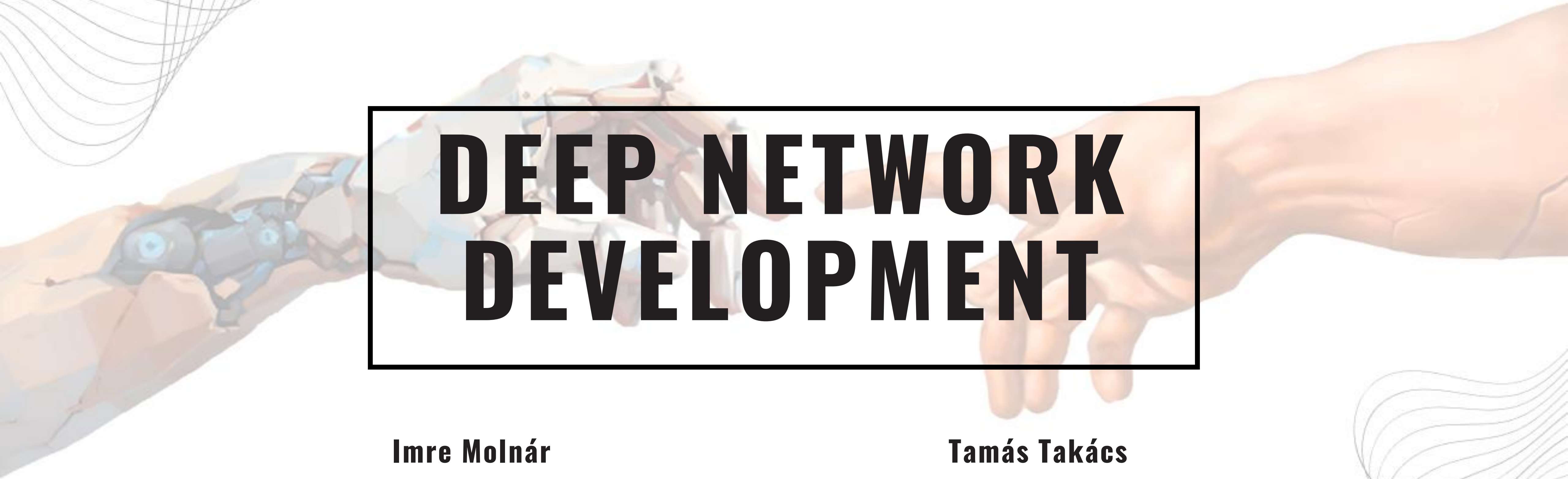


# DEEP NETWORK DEVELOPMENT

**Imre Molnár**
PhD student, ELTE, AI Department
imremolnar@inf.elte.hu
curiouspercibal.github.io

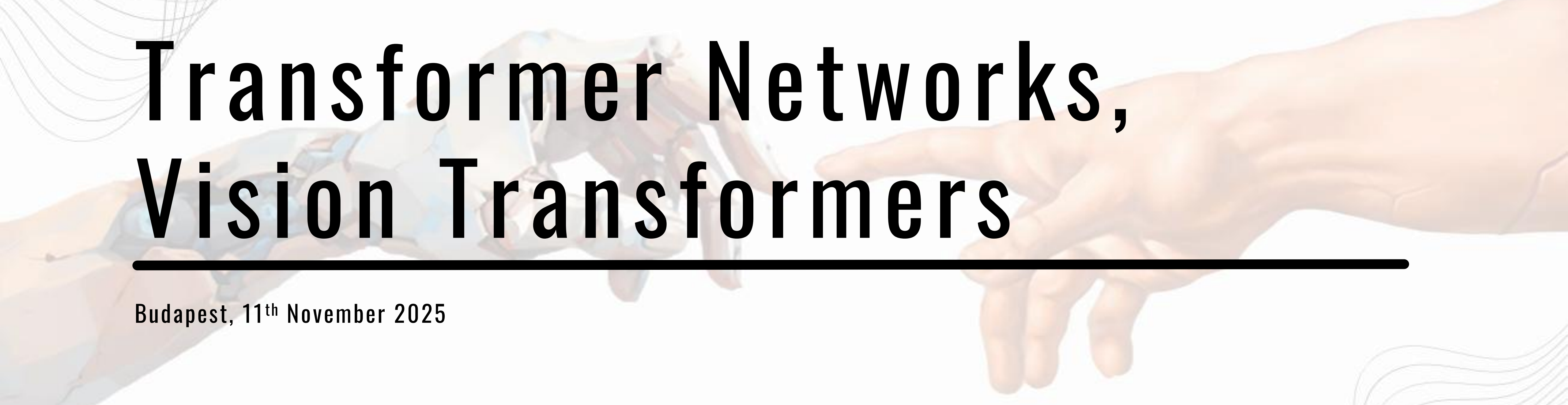
**Tamás Takács**
PhD student, ELTE, AI Department
tamastheactual@inf.elte.hu
tamastheactual.github.io

Deep Network Development

# Lecture 10.

# Transformer Networks, Vision Transformers

Budapest, 11th November 2025



# Recap

## Sequential data

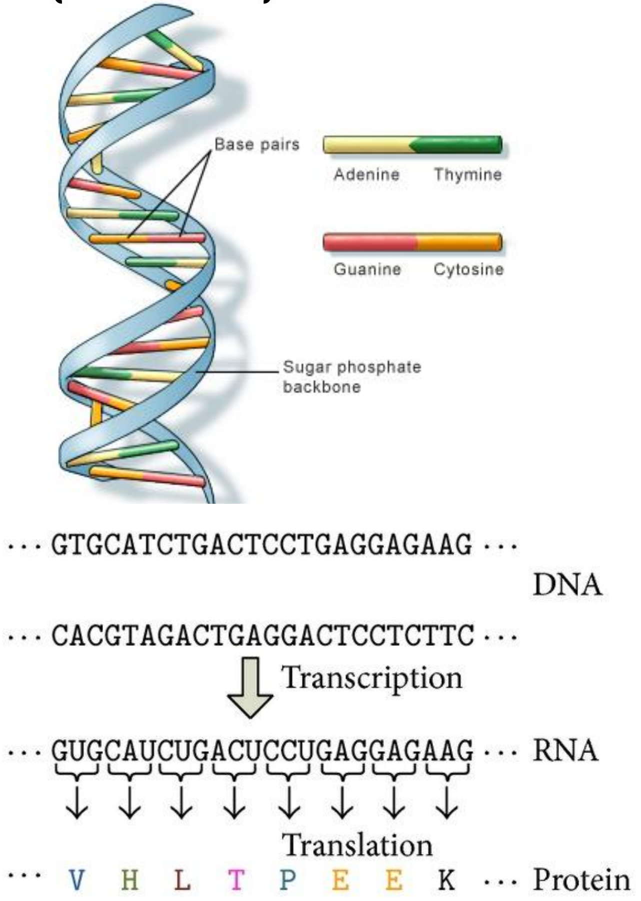
Text – sequence of words / characters

9:41
Mom
Text Message
Today 9:40 AM
Hey mom! Can you please share that lasagna recipe when you can?

Speech – sequence of signals / acoustic features

Video – sequence of images (frames)

DNA – sequence of symbols (nucleotides)


Base pairs
Adenine
Thymine
Guanine
Cytosine
Sugar phosphate backbone
··· GTGCATCTGACTCCTGAGGAGAAG ···
DNA
··· CACGTAGACTGAGGACTCCTCTTC ···
Transcription
··· GUGCAUCUGACUCCUGAGGAGAAG ··· RNA
Translation
··· V H L T P E E K ··· Protein

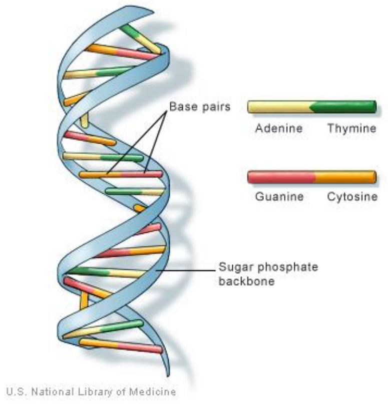
## Recap

Key properties of sequential data

- Order
- Temporal information
- Context

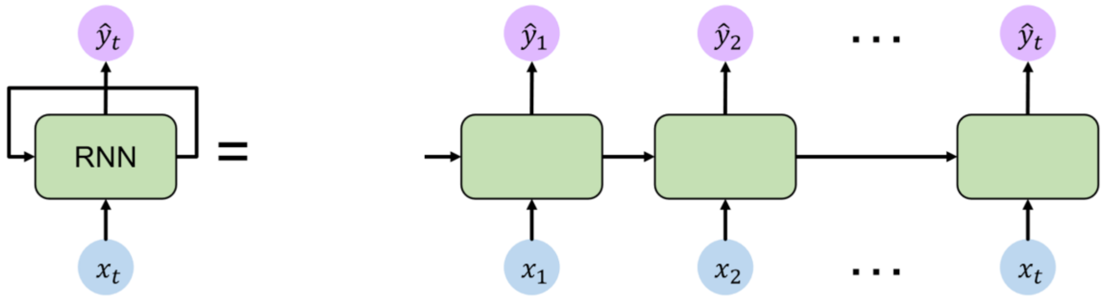
Text Message
Today 9:40 AM

Hey mom! Can you please share that lasagna recipe when you can?



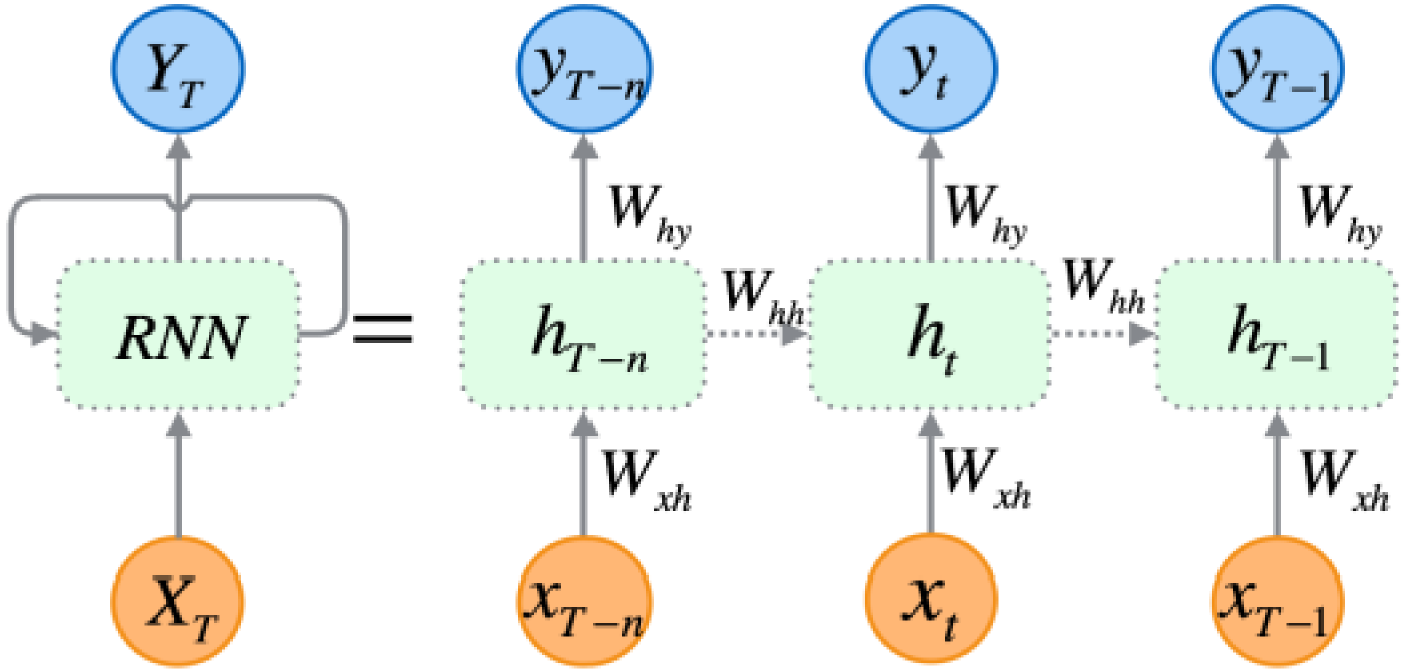
Key model properties:

- Handle variable length sequences
- Track long term dependencies
- Maintain the order of the input
- Share parameters across sequences



## Recurrent Neural Network (review)

RNN (unrolled version)



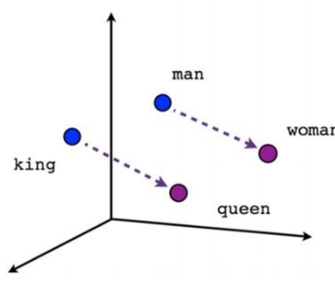
# Embeddings: Word representation (review)

## Featurized representation: Word Embedding

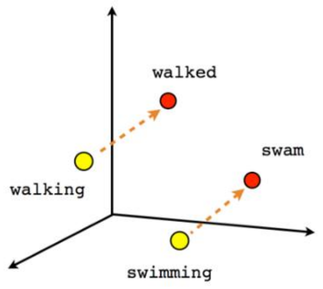
If we subtract man and woman, main difference is gender

We can compute word similarities

We can compute word analogies: man is to woman as king is to ___



Male-Female



Verb tense

| | Man (5391) | Woman (9853) | King (4914) | Queen (7151) | Apple (456) | Orange (6257) |
|---|---|---|---|---|---|---|
| Gender | **-1** | **1** | **-0.95** | **0.97** | 0.00 | 0.01 |
| Royal | 0.01 | 0.02 | **0.93** | **0.95** | -0.01 | 0.00 |
| Age | 0.03 | 0.02 | **0.7** | **0.68** | 0.03 | -0.02 |
| Food | 0.04 | 0.01 | 0.02 | 0.01 | **0.95** | **0.97** |
| … | … | … | … | … | … | … |

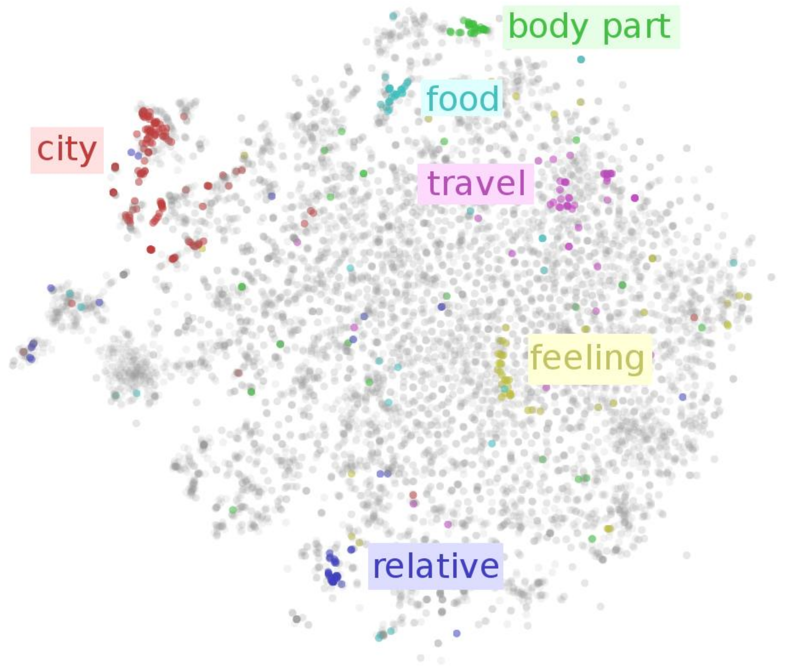
# Embeddings: Word representation (review)

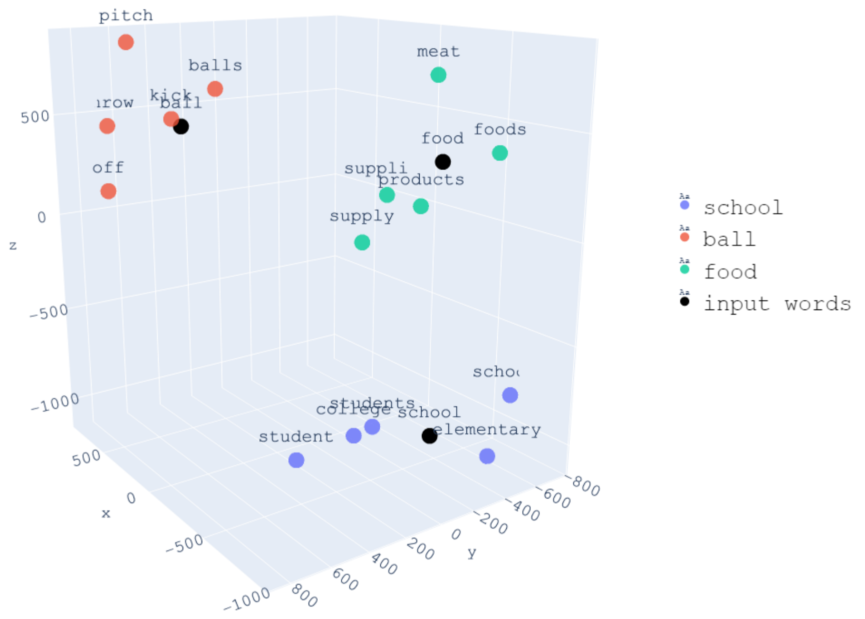
**Featurized representation: Word Embedding**

The word embeddings are learned with training.

Therefore, in practice, the features aren't that understandable.

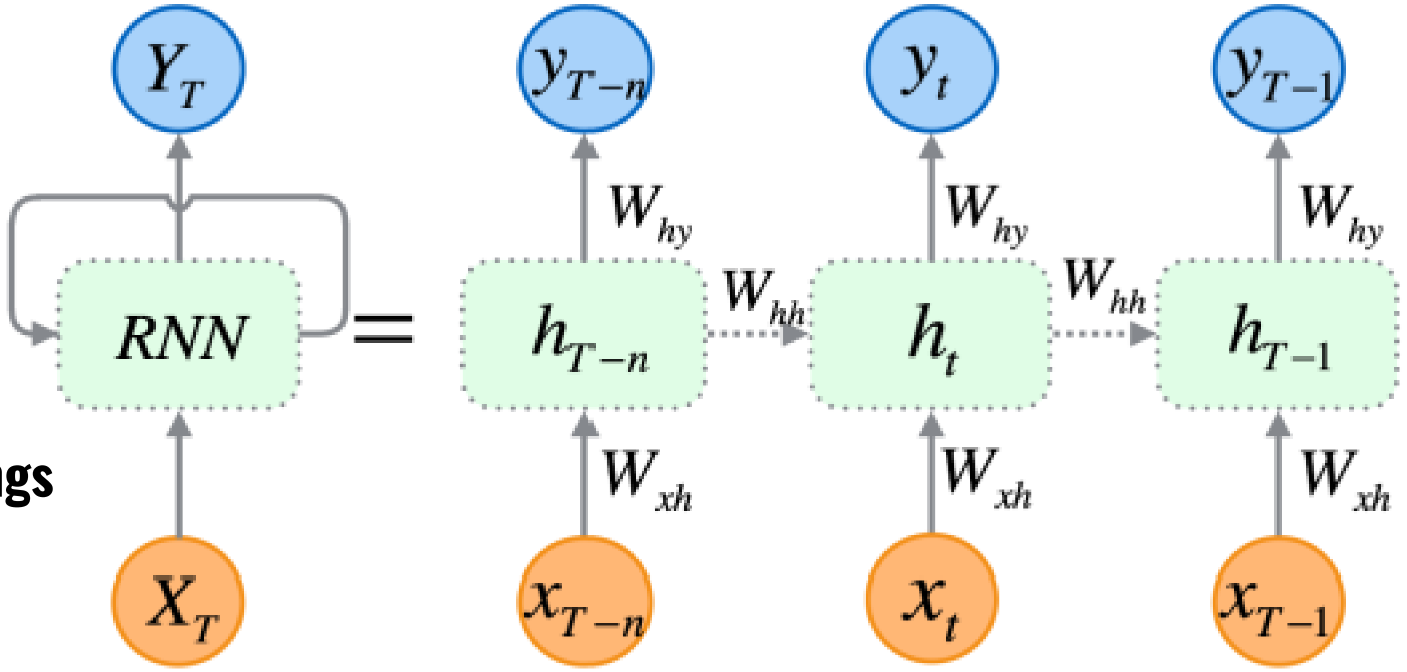
We can visualize lower representations of the embeddings with techniques such as T-SNE





## Embeddings: Word representation (review)

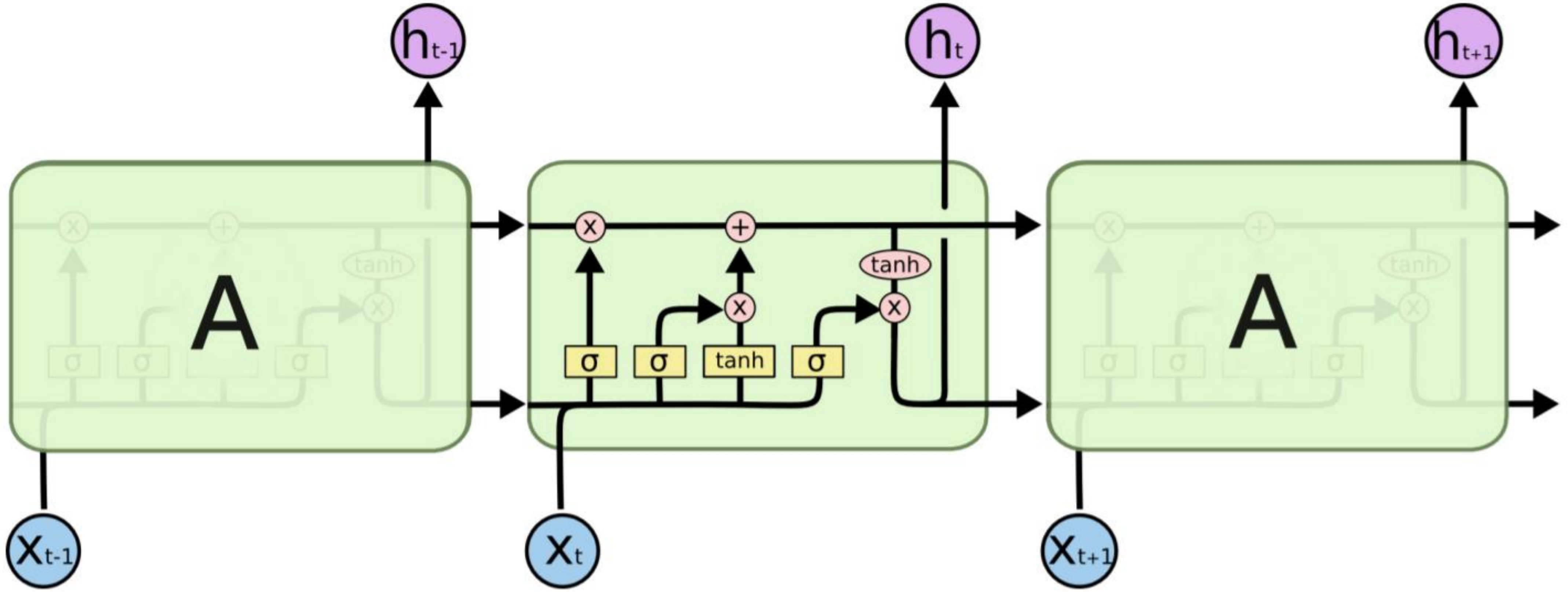
RNN (unrolled version)



**Embeddings**

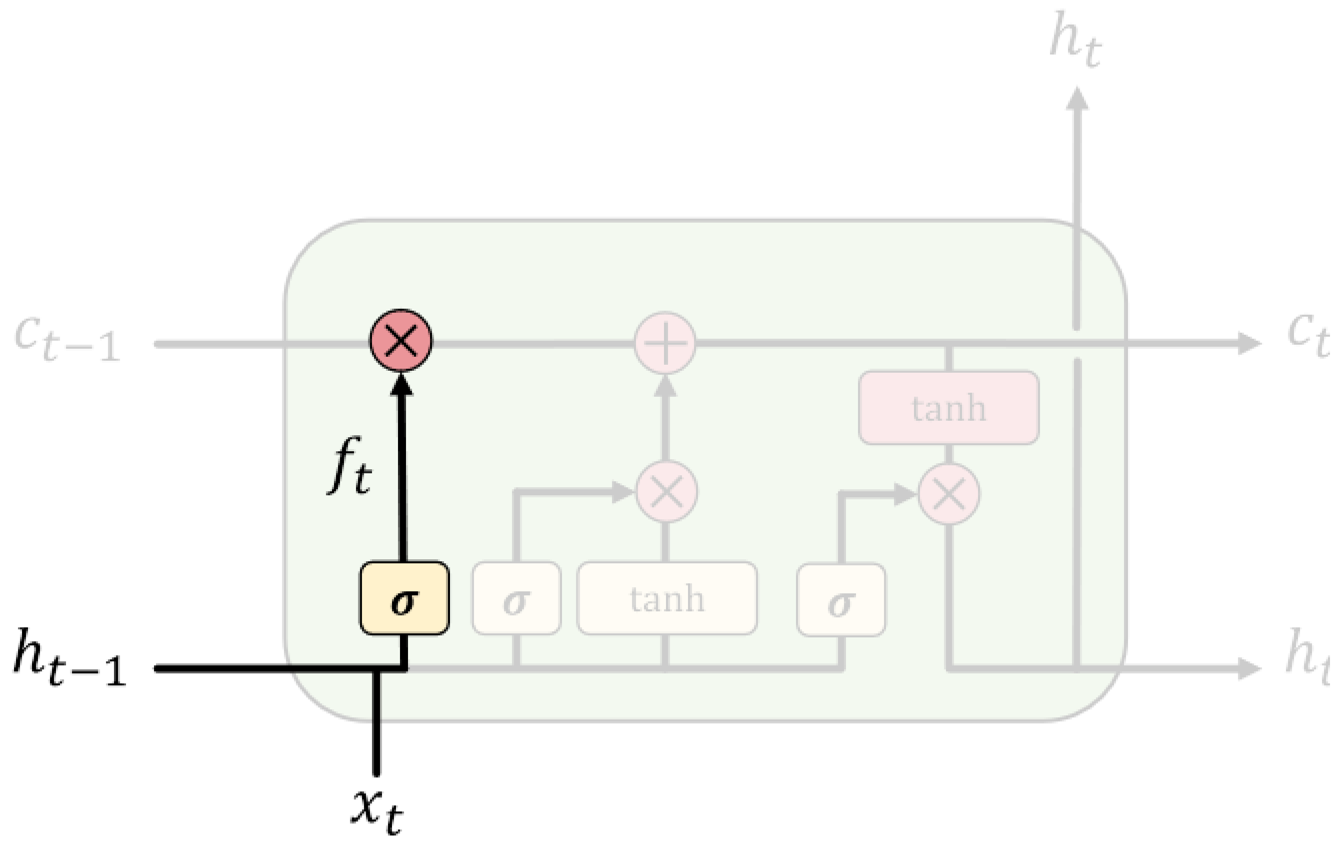
## Long Short-Term Memory (LSTM) (review)

**LSTM – a more complex network**



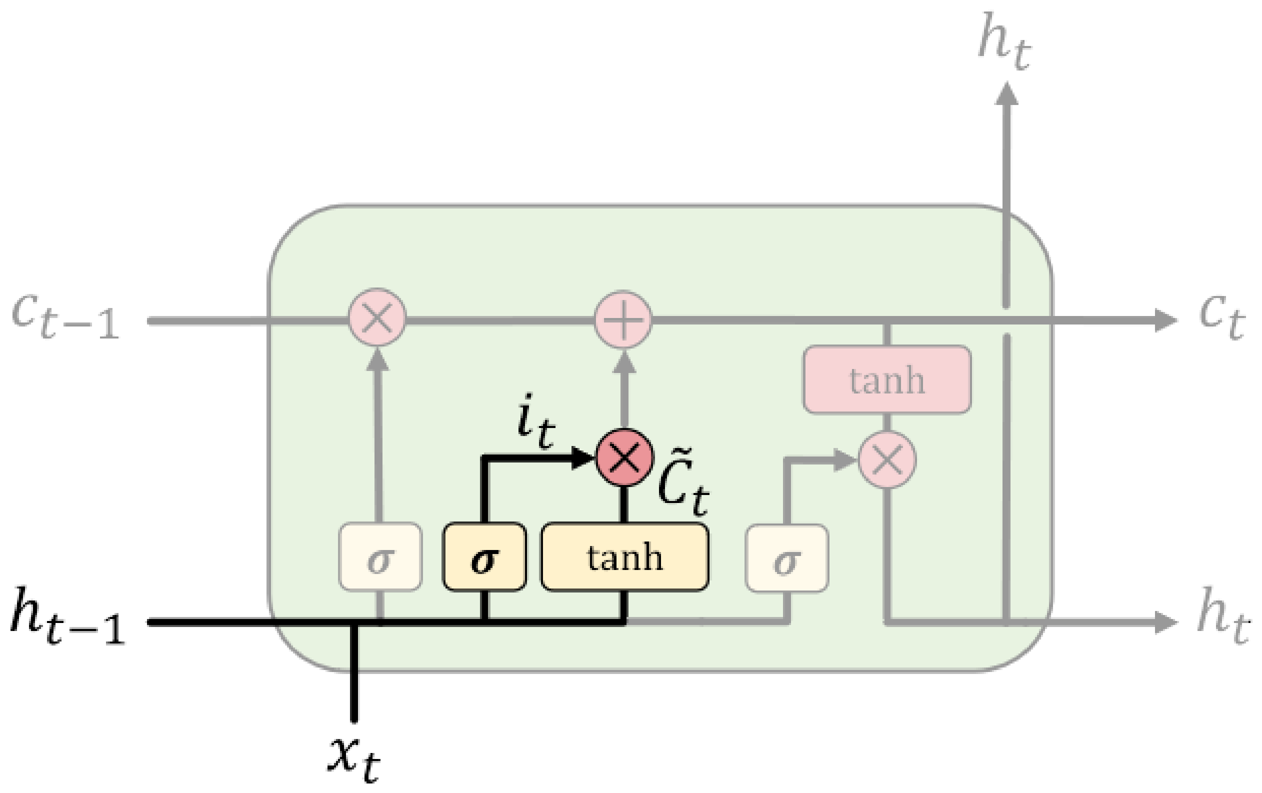
# Long Short-Term Memory (LSTM) (review)

## LSTM – Forget Gate



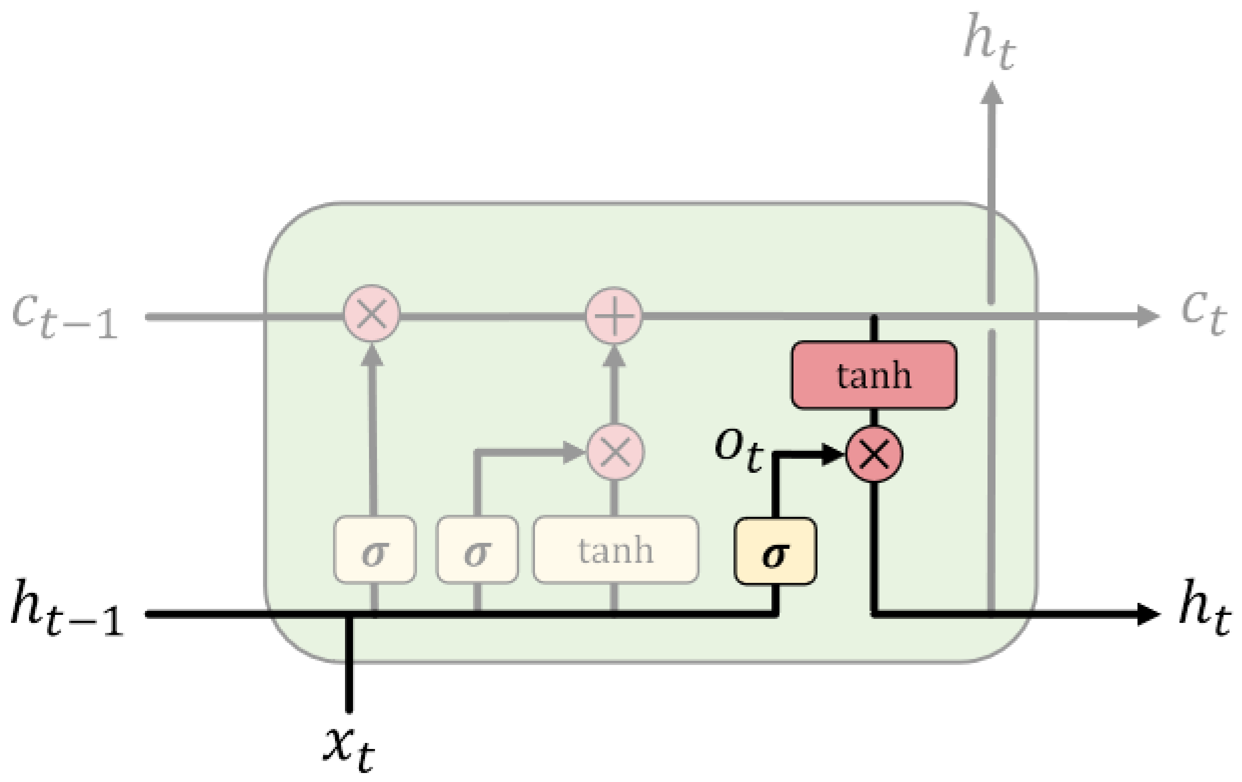
## Long Short-Term Memory (LSTM) (review)

### LSTM – Input / Ignore Gate

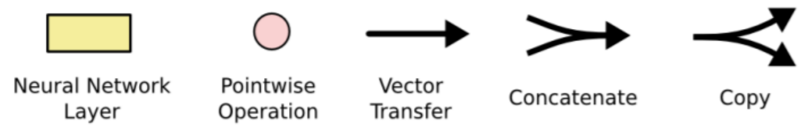


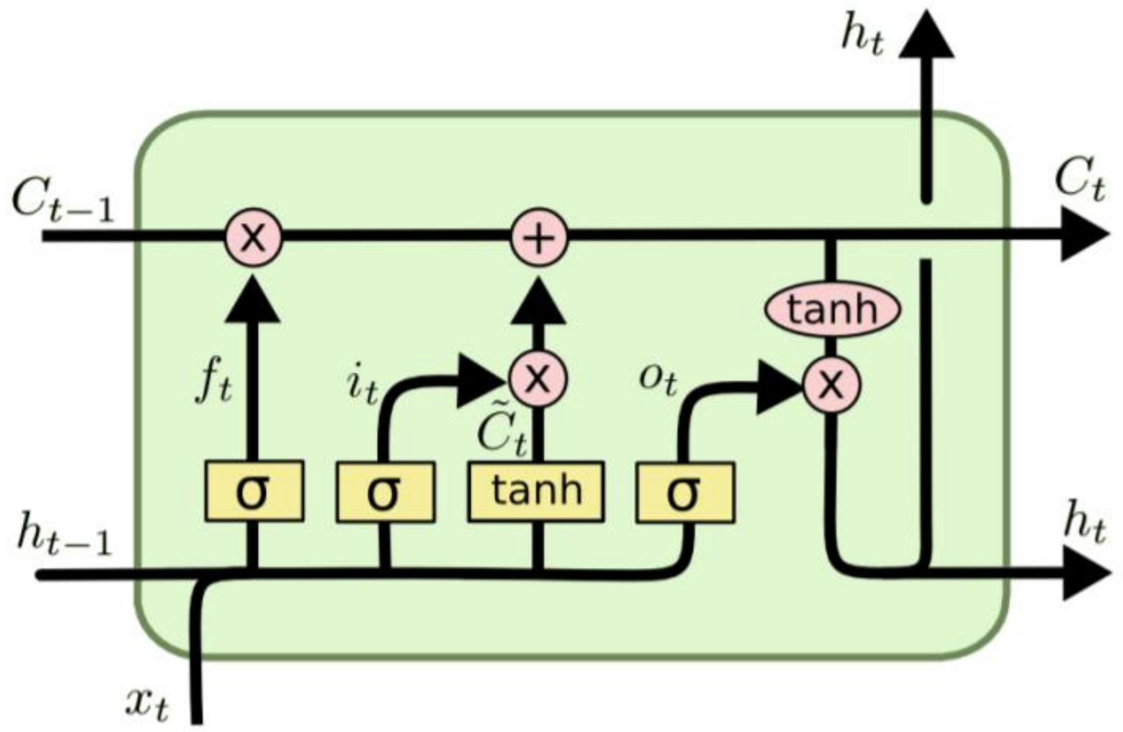
## Long Short-Term Memory (LSTM) (review)

### LSTM – Output Gate



## Long Short-Term Memory (LSTM)





**f – forget gate:** Whether to erase cell
**i – ignore gate:** Whether to write to cell
**c̃ – "vanilla RNN":** How much to write to cell
**o – output gate:** How much to reveal cell

**Gates:**

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i)$$

$$f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}x_t + b_f\right)$$

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t + b_o)$$

$$\tilde{C}_t = \tanh(W_{hc}h_{t-1} + W_{xc}x_t + b_c)$$

**Outputs:**

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

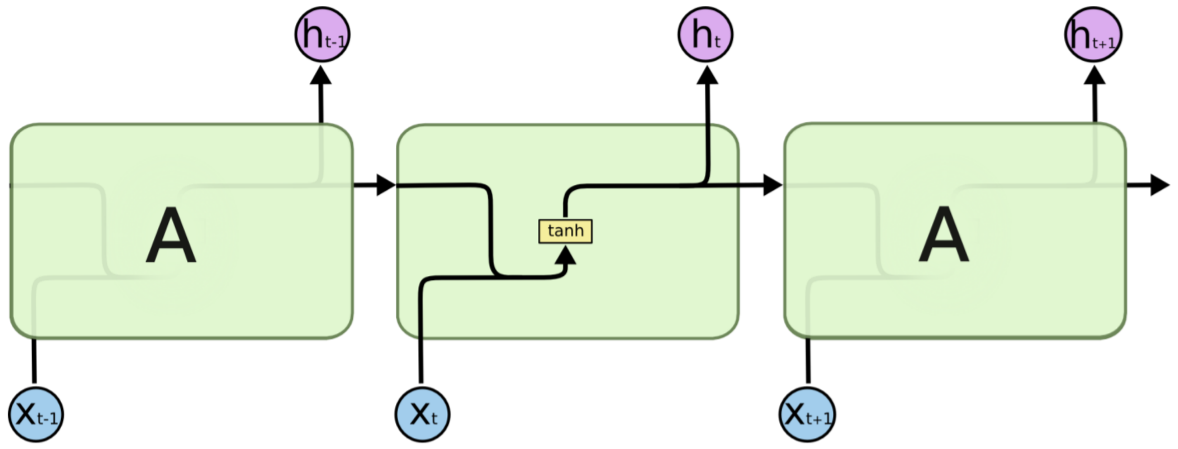
$$h_t = o_t \circ \tanh\left(\tilde{C}_t\right)$$

where $\circ$ is element-wise multiplication operation

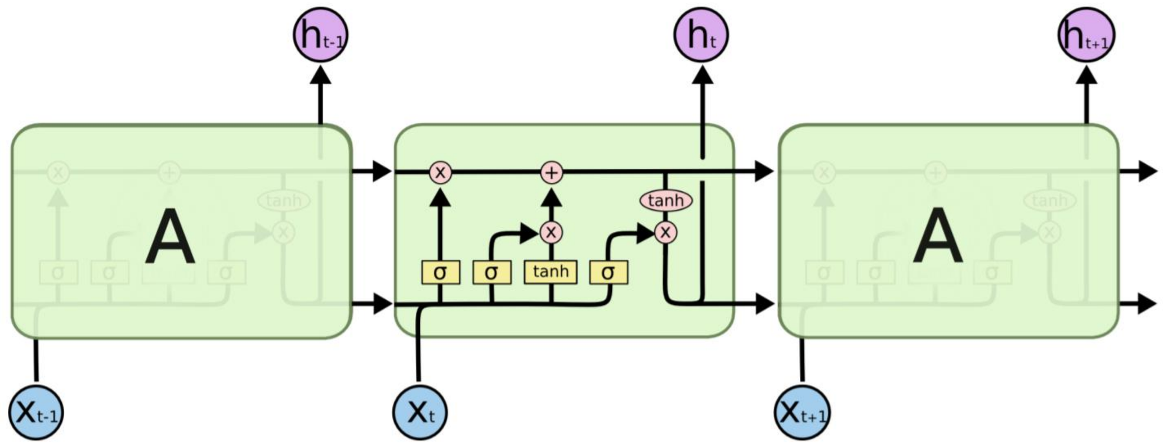
# Long Short-Term Memory (LSTM) (review)

**LSTM – a more complex network**

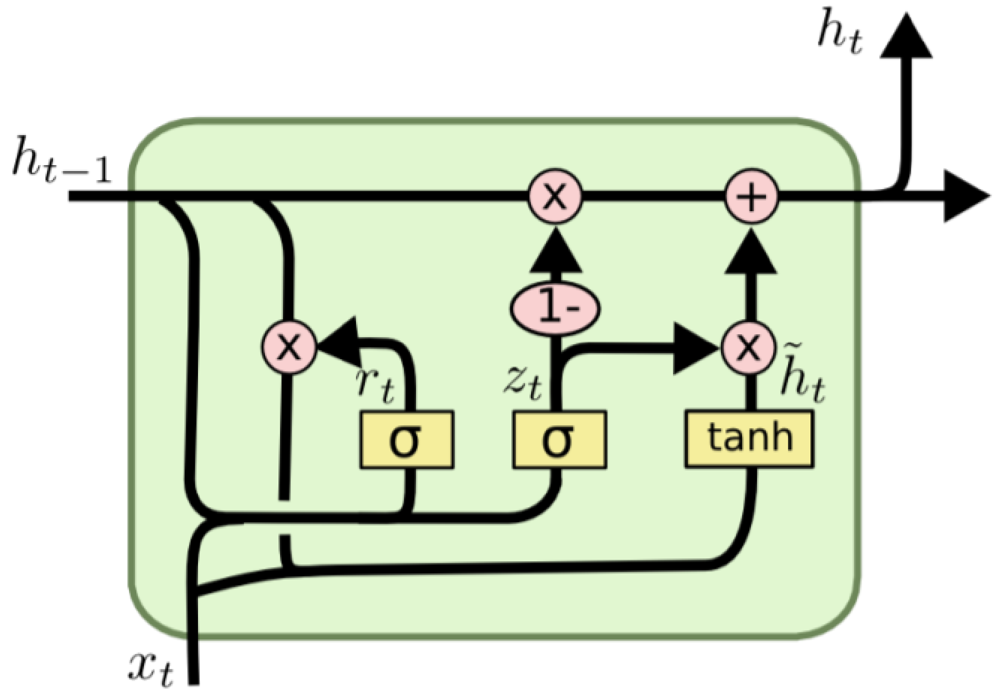
**Vanilla RNN:**

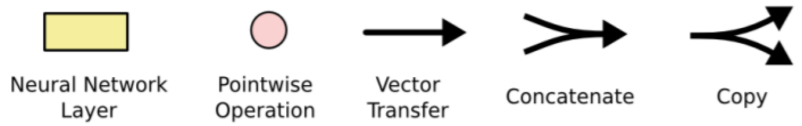


**LSTM:**



## Gated Recurrent Unit (GRU) (review)





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

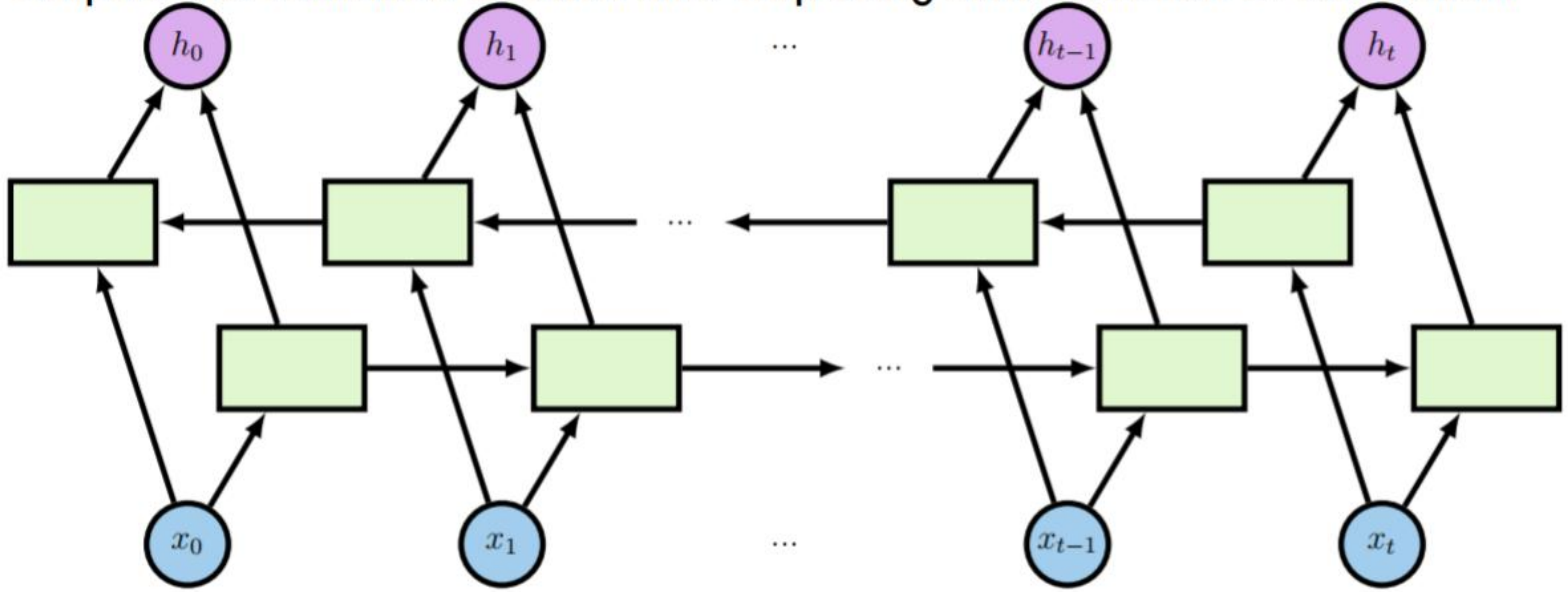
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

## Bidirectional RNN (review)

Idea: use 2 independent recurrent models together.

- Input is fed in the proper time order to the first one, and in reverse time order to the second one.
- Outputs are combined at each time step using concatenation or summation.



## Bidirectional RNN (review)

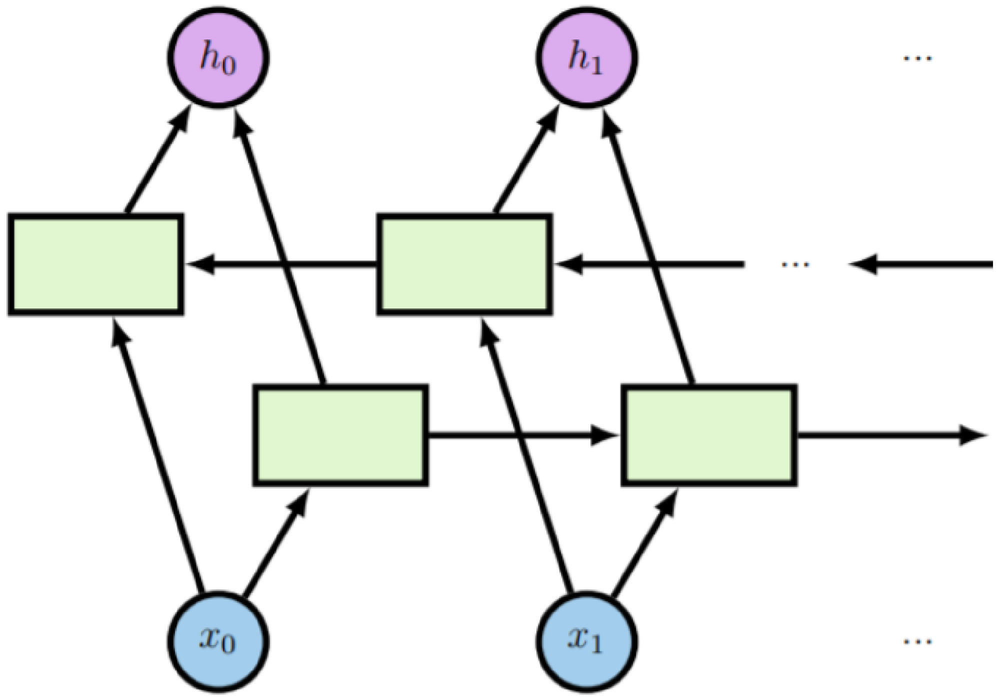
### BIDIRECTIONAL RNN (BRNN)

A BRNN layer:

$$\overrightarrow{\boldsymbol{h}}_{d,t}^{(n)} = \overrightarrow{g_d}\left(\overrightarrow{\boldsymbol{W}}_d^h \overrightarrow{\boldsymbol{h}_{d,t-1}^{(n)}} + \overrightarrow{\boldsymbol{W}}_d^x \overrightarrow{\boldsymbol{h}_{d-1,t}^{(n)}} + \overrightarrow{\boldsymbol{b}_d}\right)$$

$$\overleftarrow{\boldsymbol{h}}_{d,t}^{(n)} = \overleftarrow{g_d}\left(\overleftarrow{\boldsymbol{W}}_d^h \overleftarrow{\boldsymbol{h}_{d,t+1}^{(n)}} + \overleftarrow{\boldsymbol{W}}_d^x \overleftarrow{\boldsymbol{h}_{d-1,t}^{(n)}} + \overleftarrow{\boldsymbol{b}_d}\right)$$

$$\boldsymbol{h}_{d,t}^{(n)} = g_d\left(\overrightarrow{\boldsymbol{W}}_d \overrightarrow{\boldsymbol{h}_{d,t}^{(n)}} + \overleftarrow{\boldsymbol{W}}_d \overleftarrow{\boldsymbol{h}_{d,t}^{(n)}} + \boldsymbol{b}_d\right)$$



Where

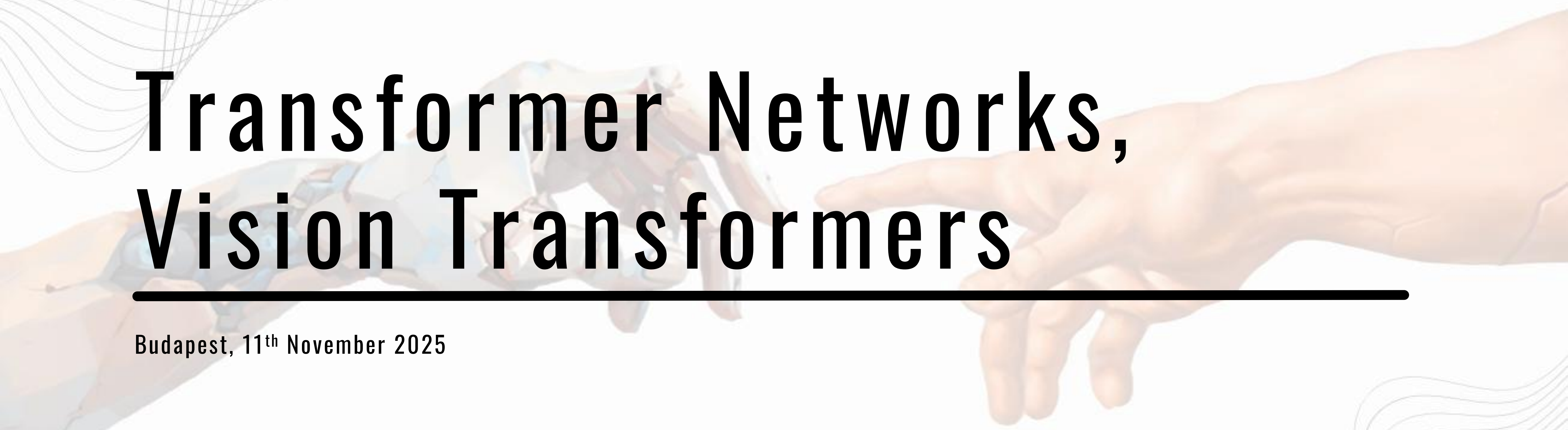
- "$\rightarrow$" means normal time order, while "$\leftarrow$" is associated with reverse time order
- $t = 1, \dots, T_n$;
- $\boldsymbol{\theta}_d = \left\{\overrightarrow{\boldsymbol{W}}_d^h, \overrightarrow{\boldsymbol{W}}_d^x, \overrightarrow{\boldsymbol{b}}_d, \overleftarrow{\boldsymbol{W}}_d^h, \overleftarrow{\boldsymbol{W}}_d^x, \overleftarrow{\boldsymbol{b}}_d, \overrightarrow{\boldsymbol{W}}_d, \overleftarrow{\boldsymbol{W}}_d, \boldsymbol{b}_d\right\}$

Deep Network Development

# Lecture 10.

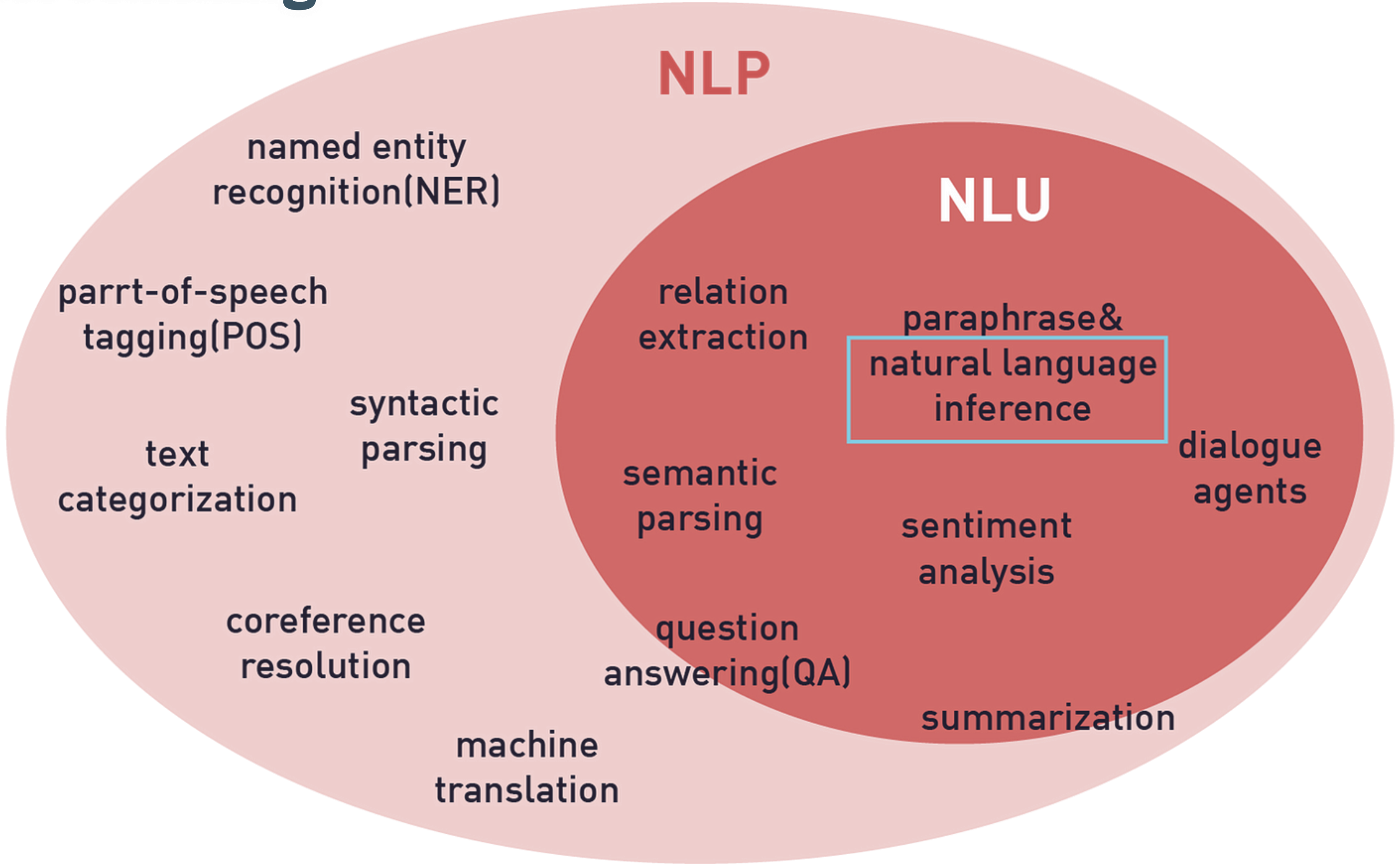
# Transformer Networks, Vision Transformers

Budapest, 11th November 2025



## Natural Language Understanding

- Neural Machine Translation
- Textual entailment
- Question answering
- Semantic similarity assessment
- Text classification
- Etc…



## Machine Translation

- Most of the proposed neural machine translation models belong to the family of **encode-decoders**
- From probabilistic perspective:

$$argmax_y p(y \mid x)$$

- Which means finding a target **y** that maximizes the conditional probability of **y** given a source sentence **x**

Machine Translation

## Machine Translation

- In the **encoder-decoder** framework an **encoder** reads and reduces the input sentence (a sequence of vectors $\mathbf{x} = (x_1, \dots, x_T)$ ) into a vector $c$
- The approach for RNNs:

$$h_t = f(x_t, h_{t-1})$$
$$c = q\left(\{h_1, \dots, h_{T_x}\}\right)$$

- Where $q$ is a nonlinear function, $h_t$ is the hidden state at time $t$
- If we choose $f$ as an LSTM then $q\left(\{h_1, \dots, h_{T_x}\}\right) = h_{T_x}$
- The decoder then predicts the next word $y_t$ given the context vector $c$ and all the previously predicted words $\{y_1, \dots, y_{t-1}\}$

Machine Translation

## Machine Translation

- The **decoder** then predicts the next word $y_t$ given the context vector $c$ and all the previously predicted words $\{y_1, \dots, y_{t-1}\}$
- The predicted translation $\mathbf{y}$ can be expressed as the joint probability of:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \dots, y_{t-1}\}, c)$$

- where $\mathbf{y} = \{y_1, \dots, y_{T_y}\}$
- Whit and RNN this is modelled as:

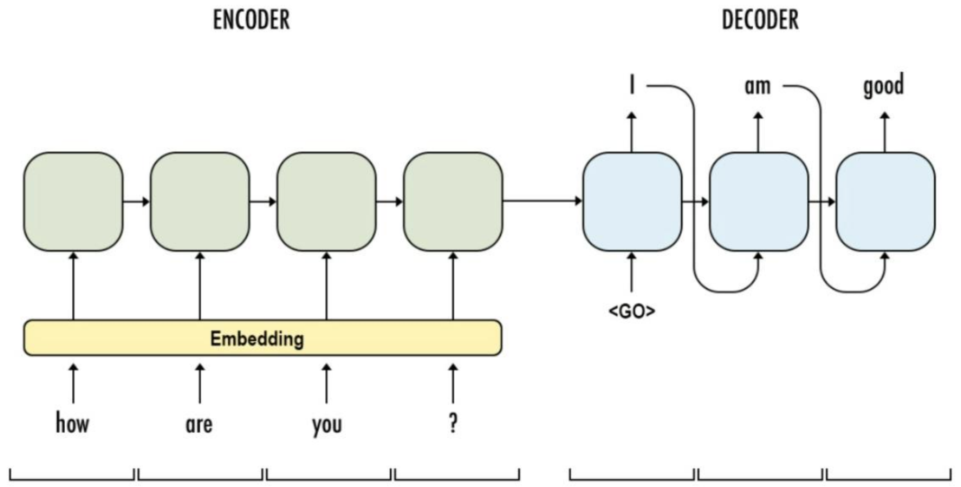
$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

- Where $g$ nonlinear multi layered function and $s_t$ is the hidden state

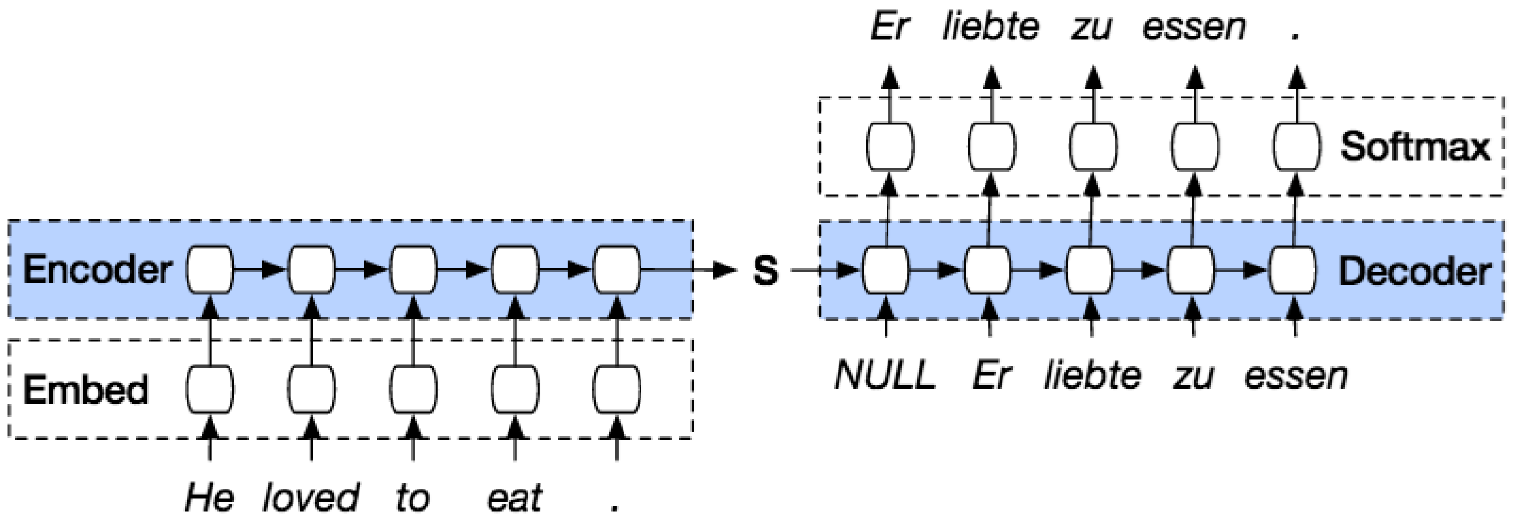
Machine Translation

## Encoder – Decoder (Seq2Seq)

Question : Answer
(sequence) : (sequence)

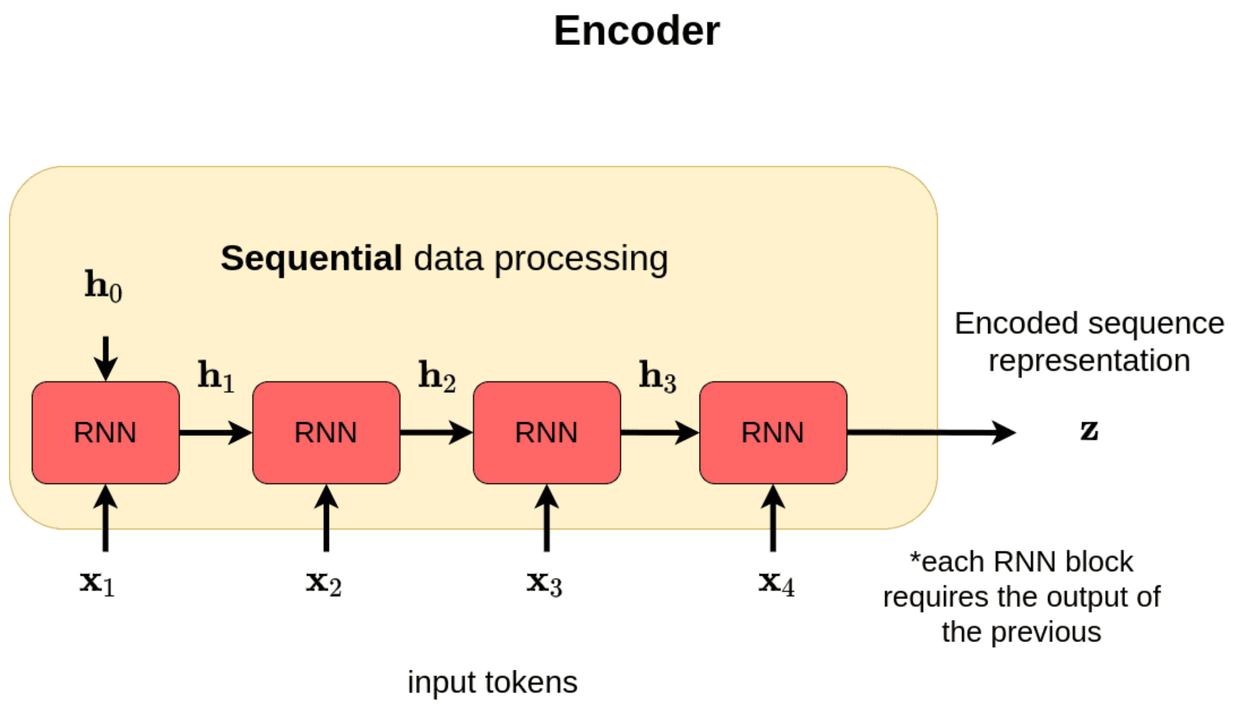


Machine Translation
English : German
(sequence) : (sequence)



## Encoder – Decoder (Seq2Seq)

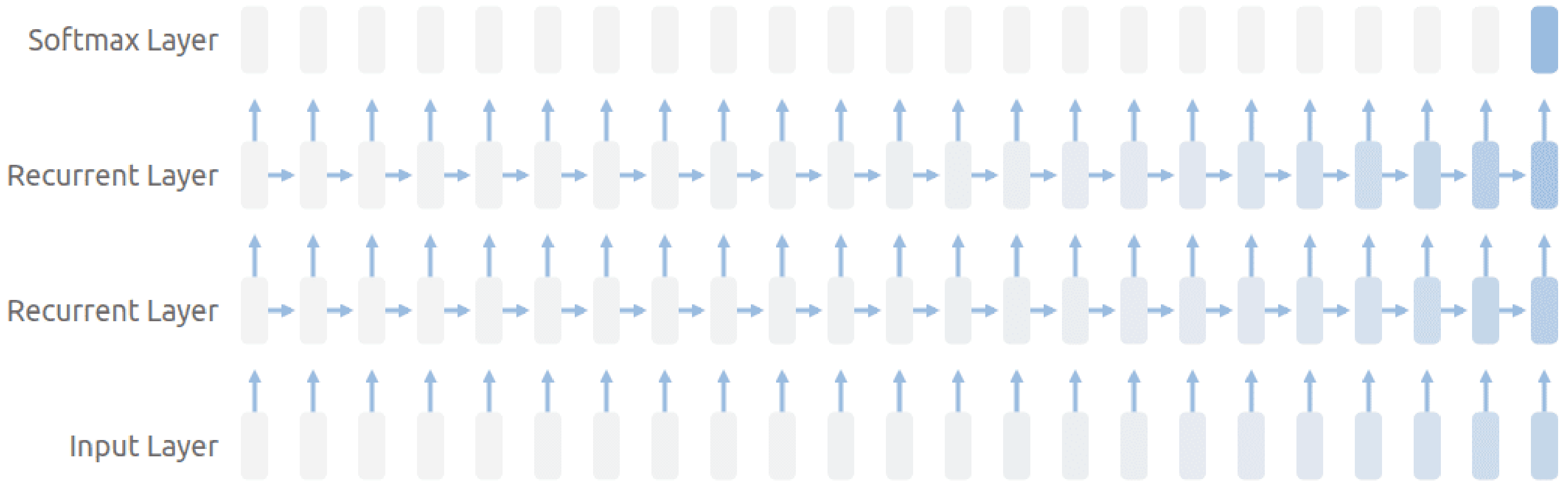
- The encoder and decoder are nothing more than stacked RNN layers, such as LSTM's. The encoder processes the input and produces one compact representation, called z, from all the input timesteps. It can be regarded as a compressed format of the input.



## Encoder – Decoder (Seq2Seq)

**Problems with this approach**

- The intermediate representation **z** cannot encode information from all the input timesteps. This is commonly known as the **bottleneck problem**. The vector z needs to capture all the information about the source sentence.
- In practice, how far we can see in the past (the so-called reference window) is finite. RNN's tend to **forget information** from timesteps that are far behind.

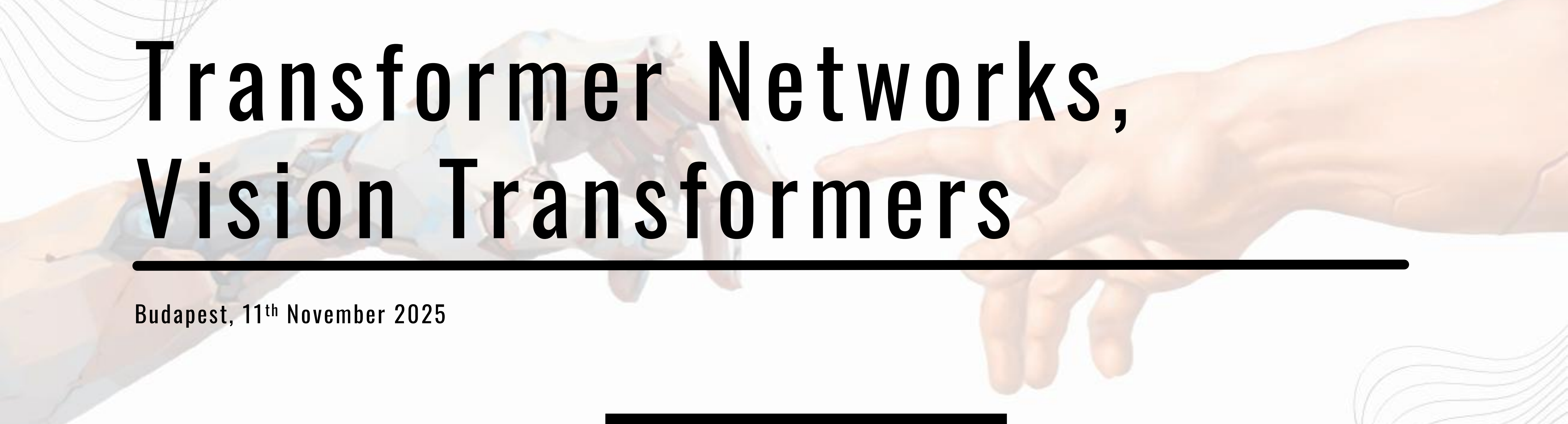


**Vanishing Gradient:** where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit.

Deep Network Development

# Lecture 10.

# Transformer Networks, Vision Transformers



Budapest, 11th November 2025

1 Natural Language Understanding

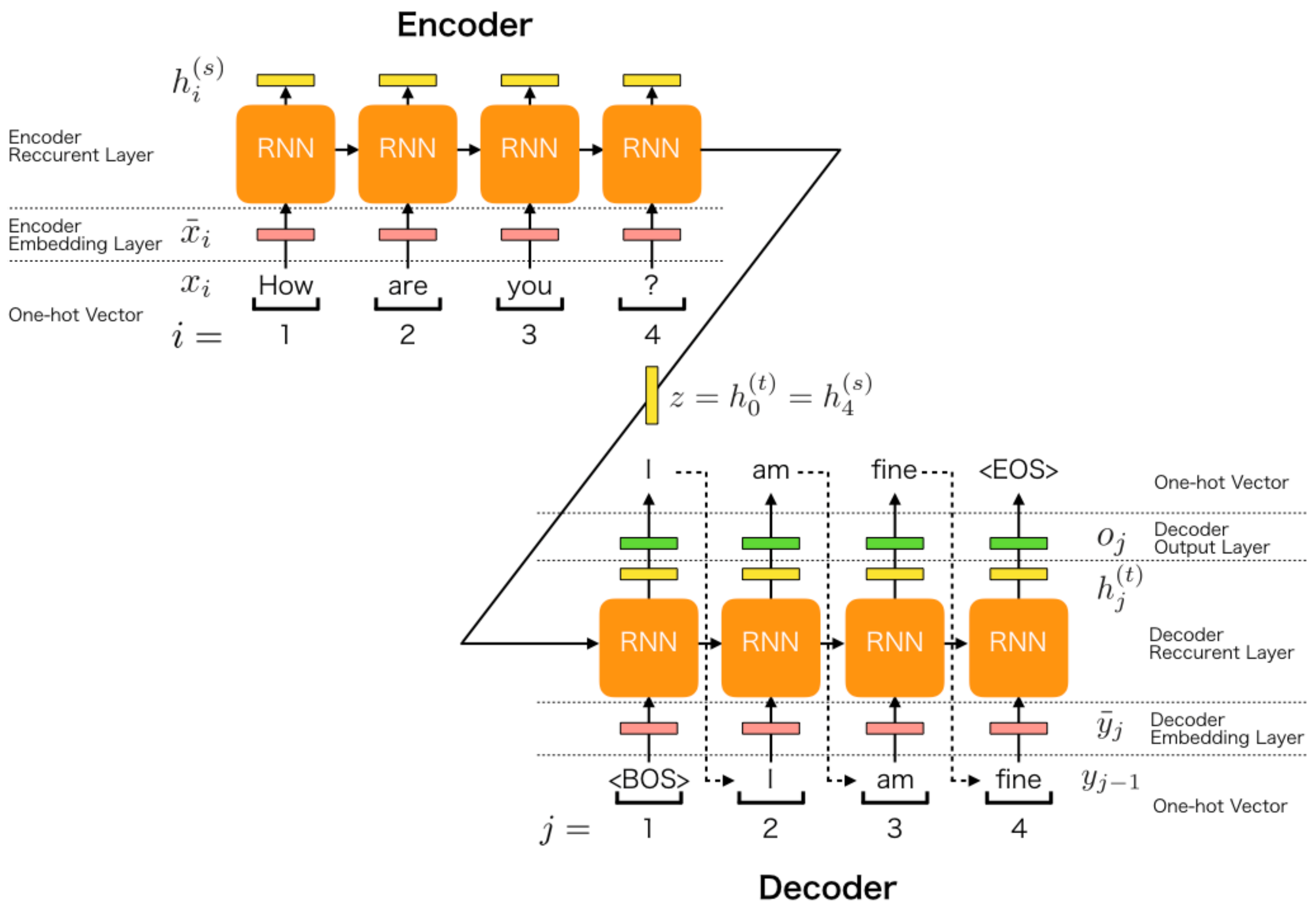
**2 Attention Mechanism**

3 Transformer Architecture

4 Vision Transformers

## Introduction to Attention Mechanism

- All the information from the encoder is represented in the z vector (context)
- However, as seen previously, information from earlier timestamps is not preserved
- Can we create a better context vector?



## Introduction to Attention Mechanism

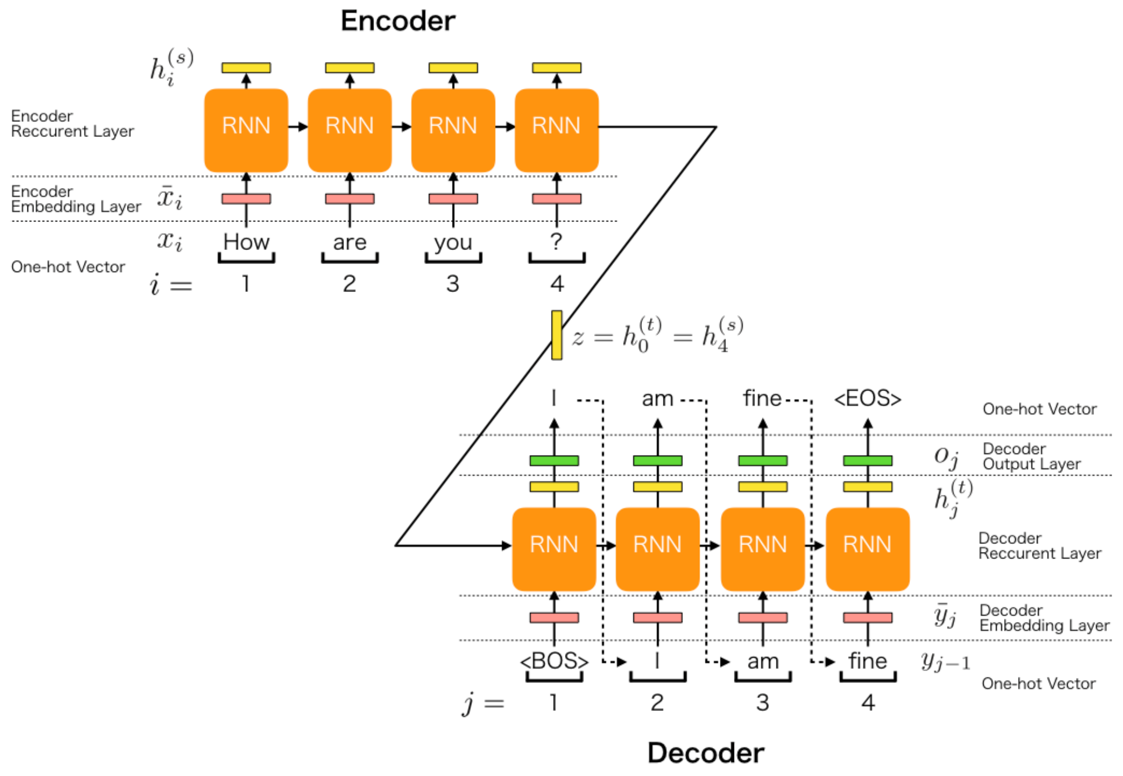
- In the new approach, we define each conditional probability as:

$$p(y_i \mid \{y_1, \dots, y_{i-1}\}, c) = g(y_{i-1}, s_i, c_i)$$

- Where $s_i$ is an RNN hidden state for time $i$, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- Note that here we produce a distinct context vector $c_i$ for each target word $y_i$



## Introduction to Attention Mechanism (review)

- The core idea is that the context vector $c_i$ should have access to all parts of the input sequence instead of just the last one.
- In other words, we need to form a **direct connection** with each timestamp. We can look at all the different words at the same time and learn to "pay attention" to the correct ones depending on the task at hand.
- In the encoder-decoder:
  - Given the hidden states of the encoder at each time step $h = \{h_1, \dots, h_{T_x}\}$
  - Given the previous state in the decoder $y_{i-1}$
  - We define an attention network that gives the attention scores for the current state of the decoder

$$e_{ij} = \text{attention}_{\text{net}}(y_{i-1}, \mathbf{h}) \in \mathbb{R}^n$$

  - We convert this scores into probabilities

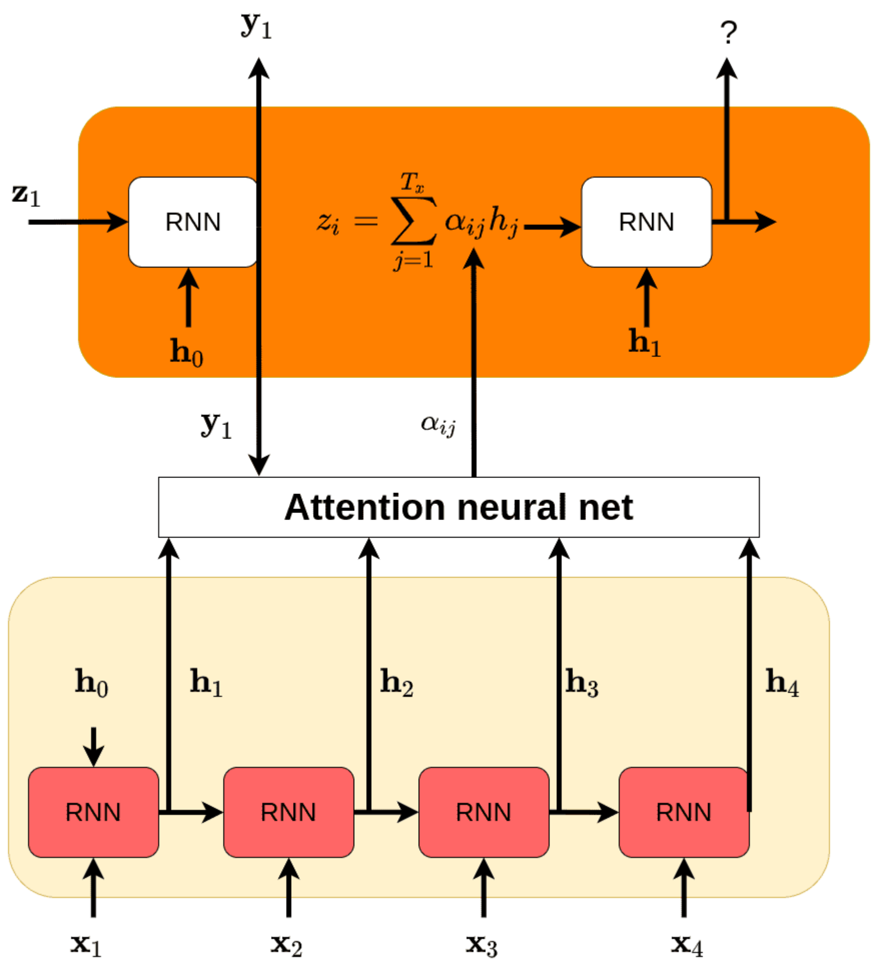
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

  - Finally, we get our new context vector z:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

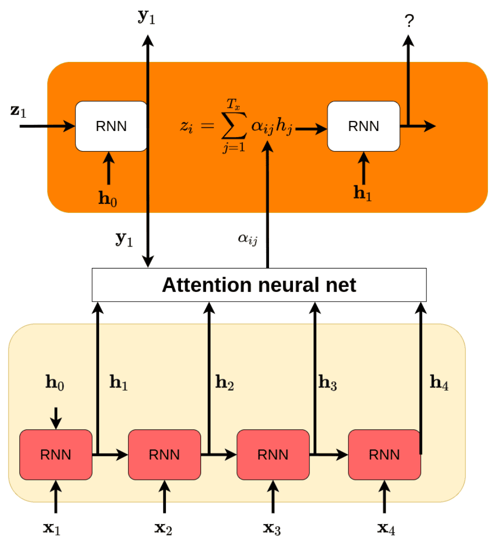
## Introduction to Attention Mechanism (review)

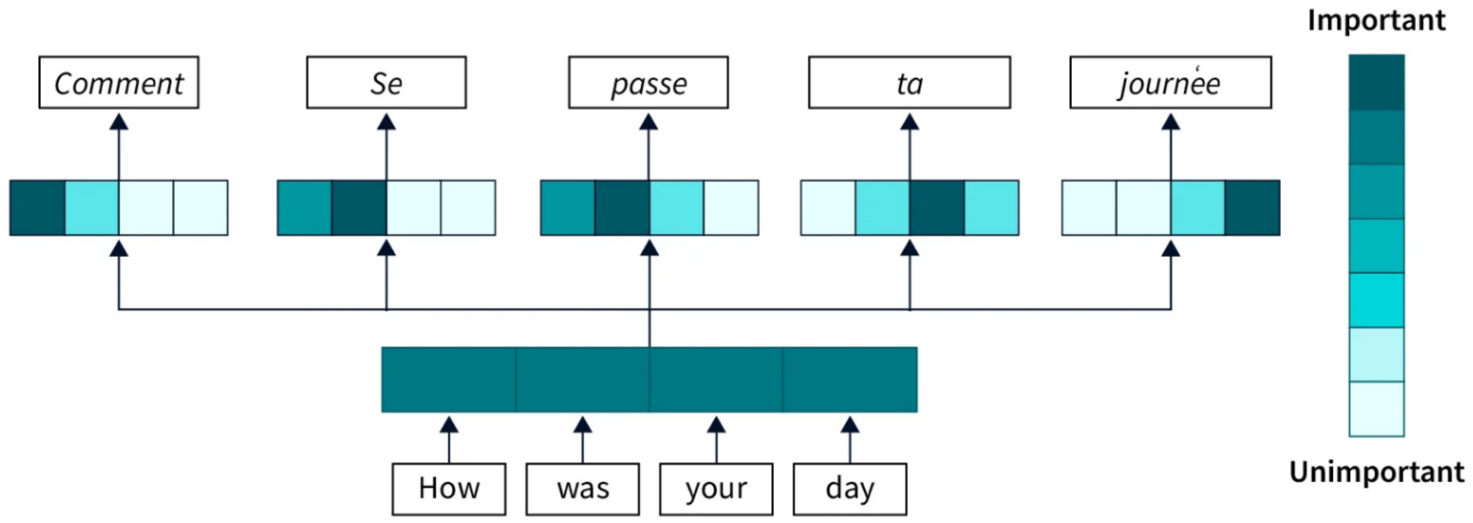
- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



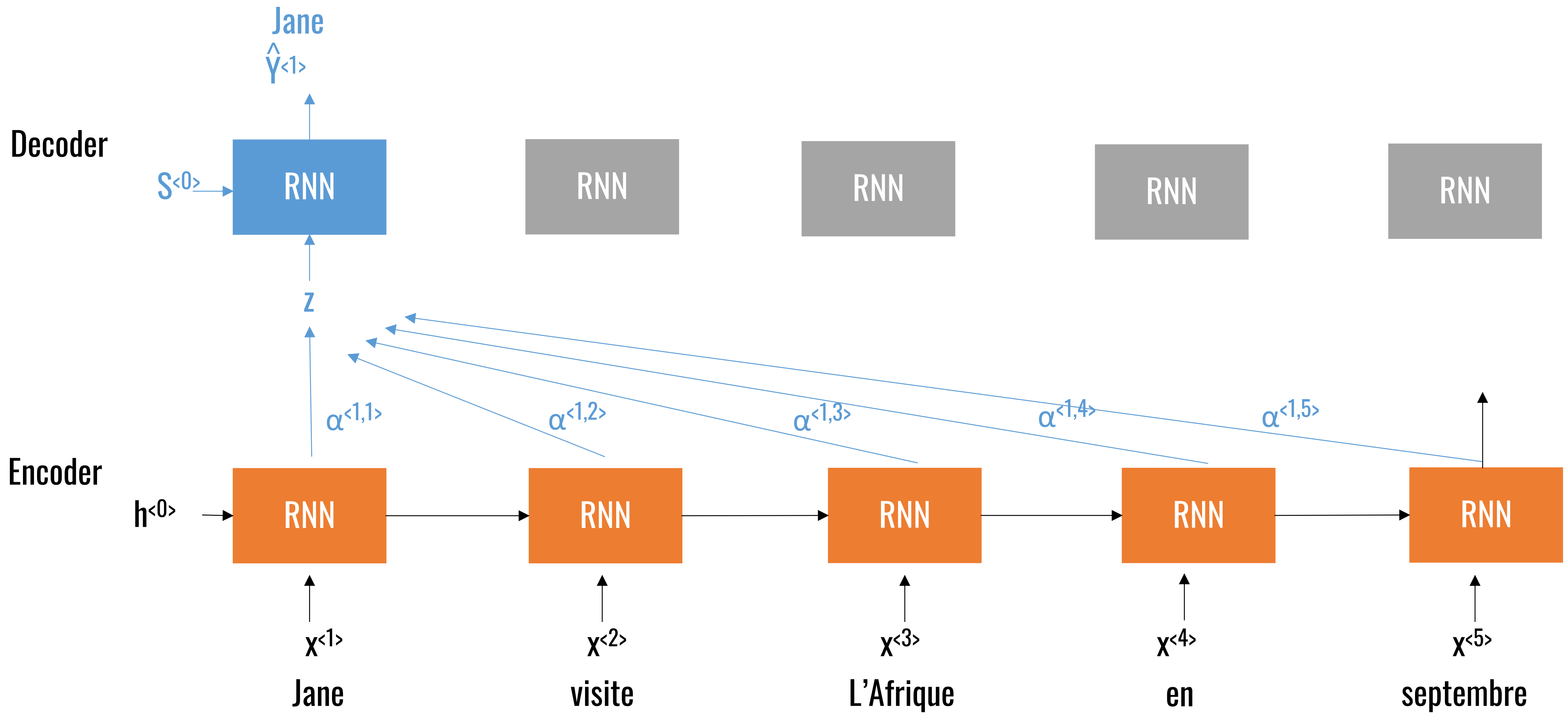
## Introduction to Attention Mechanism (review)

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder





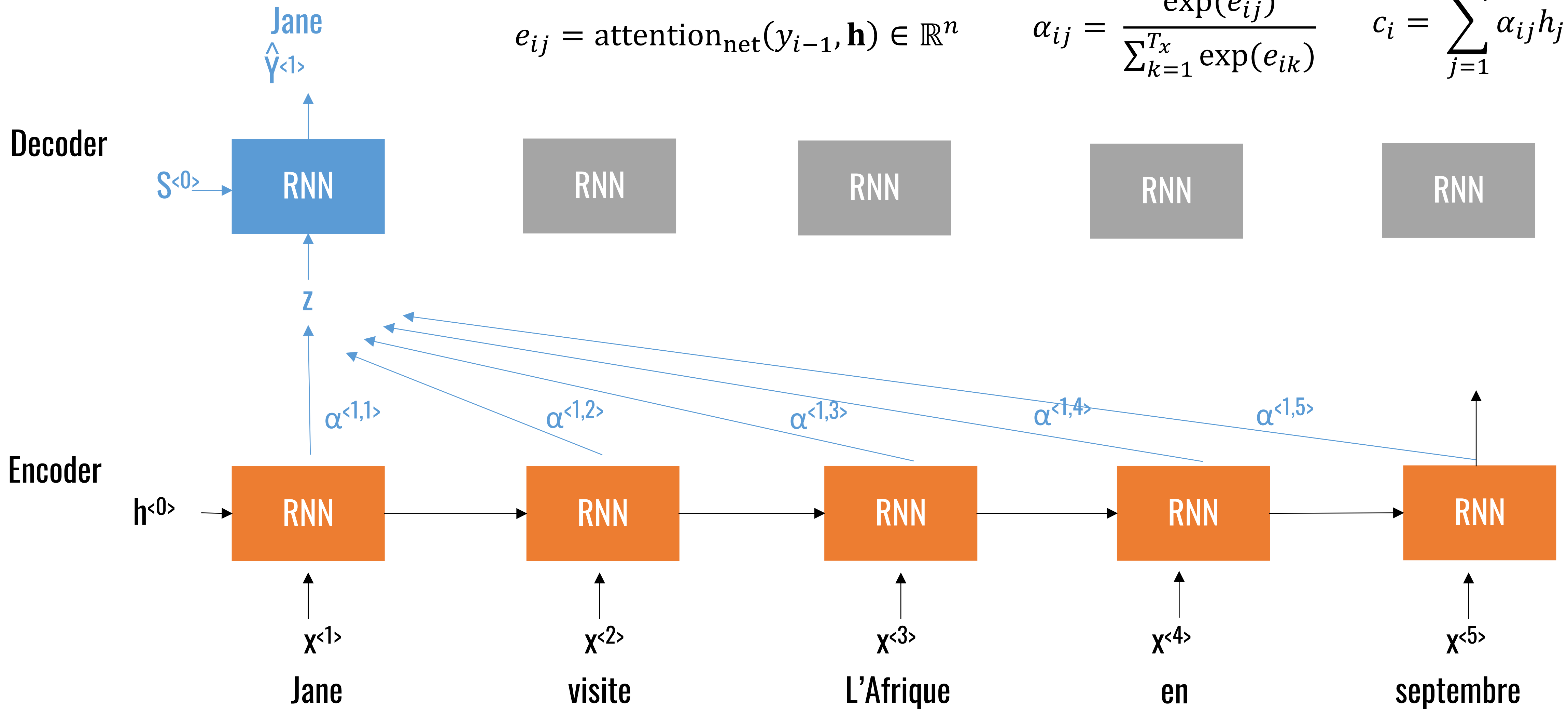
## Attention Mechanism (review)



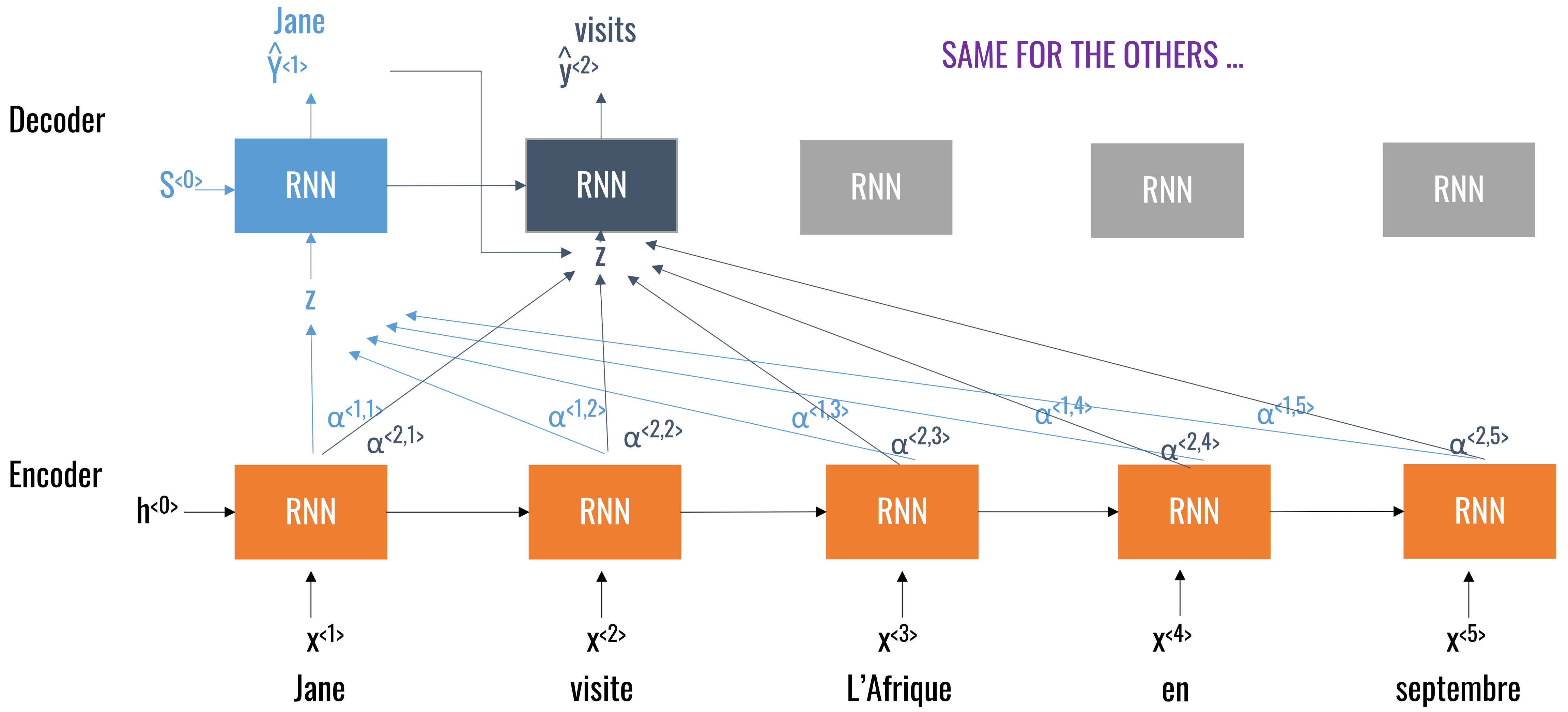
# Attention Mechanism (review)

$\alpha_{ij}$ = amount of "attention" $y_i$ should pay to $h_*$

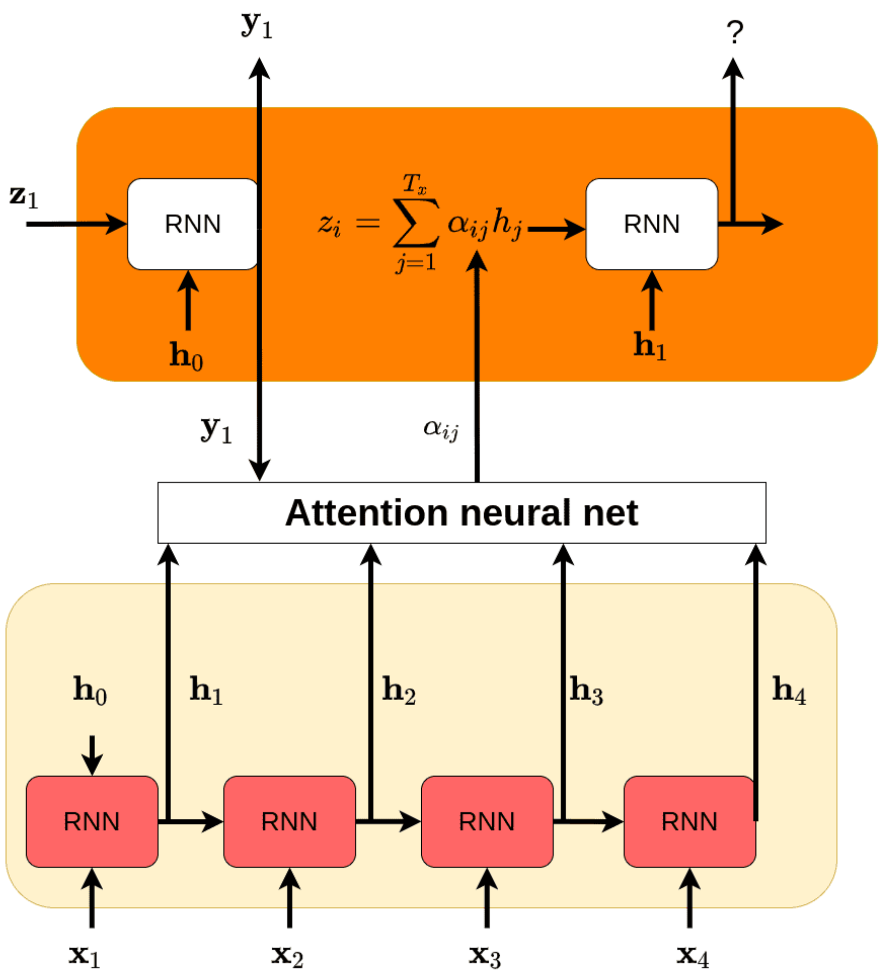
$$e_{ij} = \text{attention}_{\text{net}}(y_{i-1}, \mathbf{h}) \in \mathbb{R}^n \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

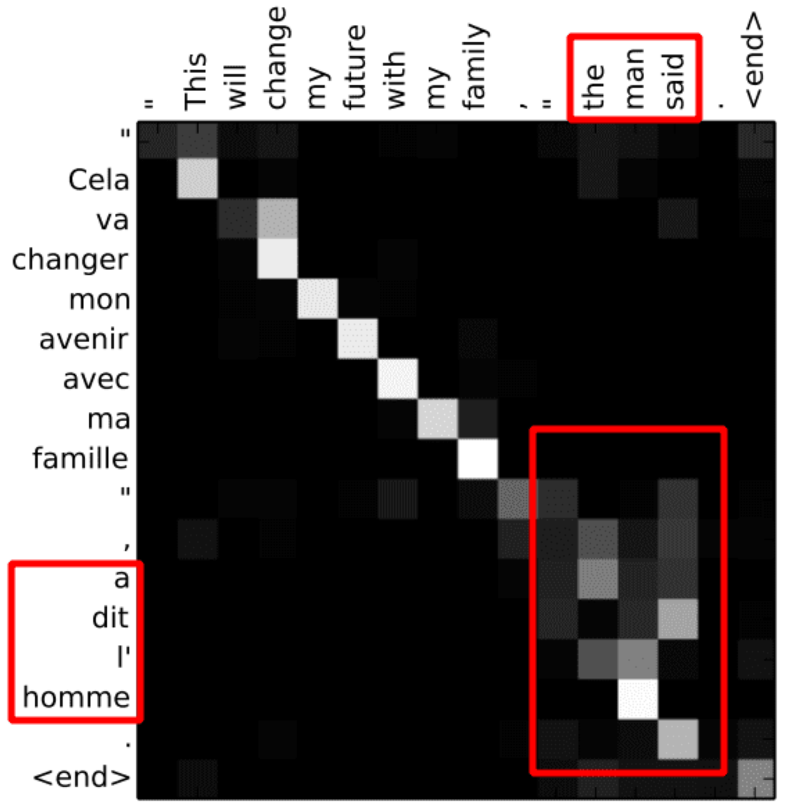


## Attention Mechanism (review)

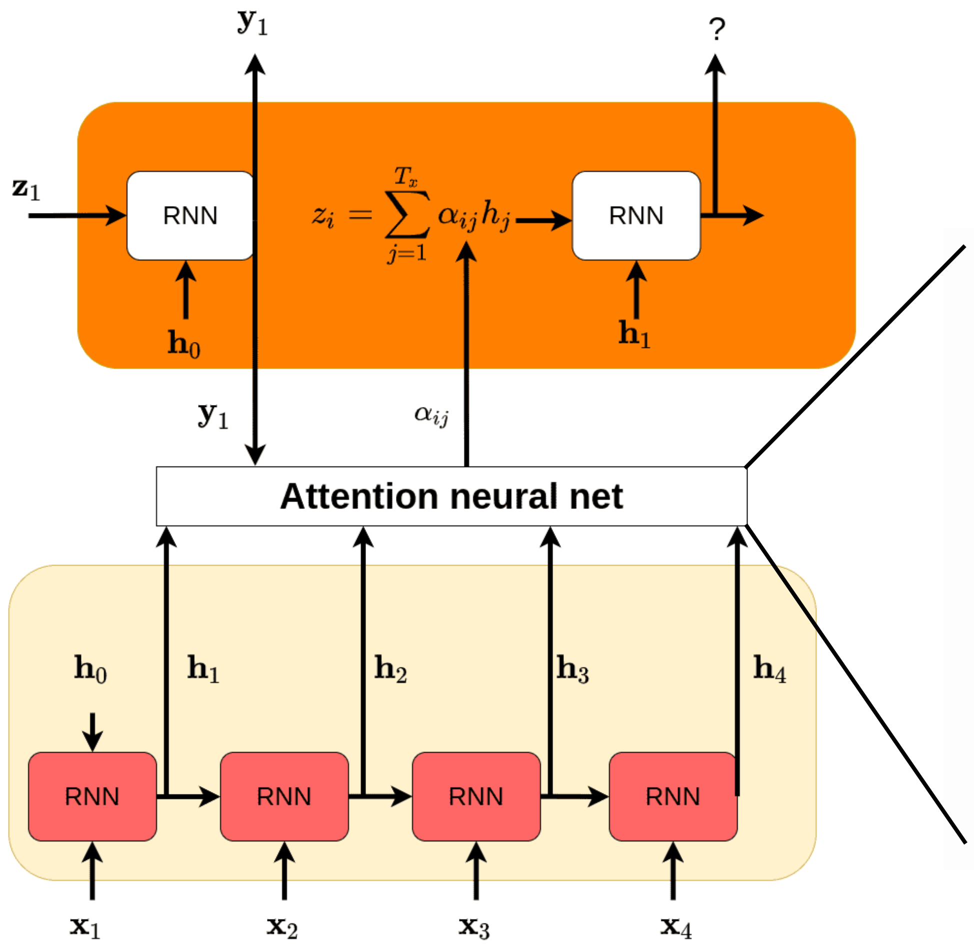


## How does Attention work?

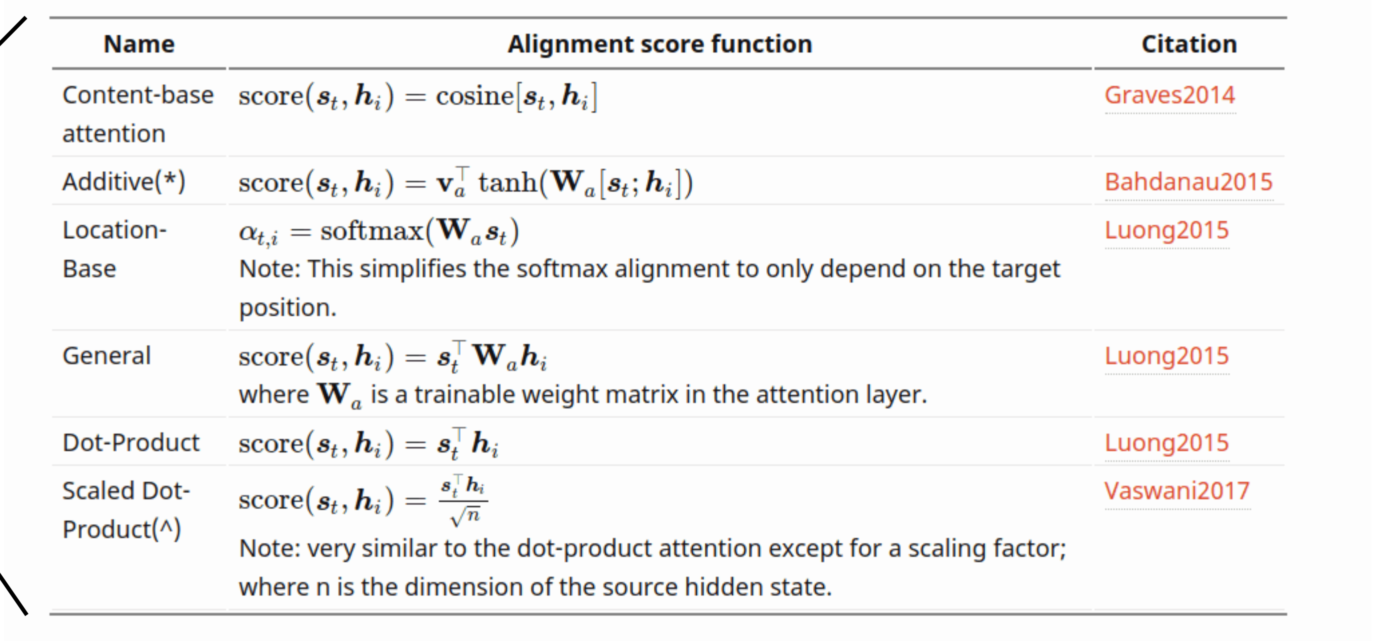




## Attention Mechanism



Several ways to calculate the scores

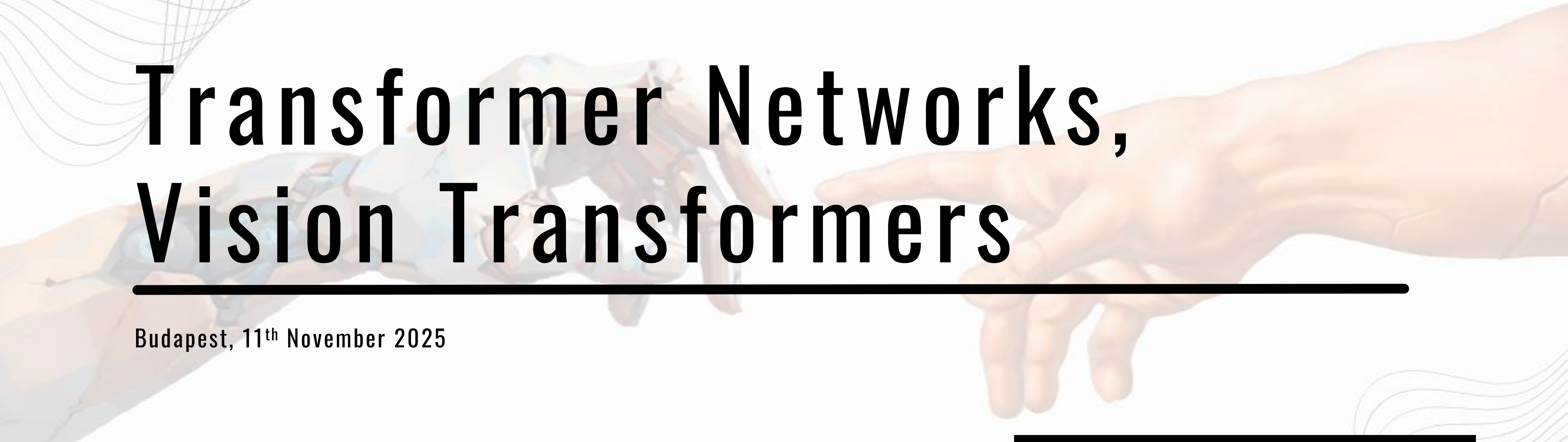


| Name | Alignment score function | Citation |
|---|---|---|
| Content-base attention | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$ | Graves2014 |
| Additive(*) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$ | Bahdanau2015 |
| Location-Base | $\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$<br>Note: This simplifies the softmax alignment to only depend on the target position. | Luong2015 |
| General | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$<br>where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer. | Luong2015 |
| Dot-Product | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$ | Luong2015 |
| Scaled Dot-Product(^) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$<br>Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state. | Vaswani2017 |

Deep Network Development

# Lecture 10.

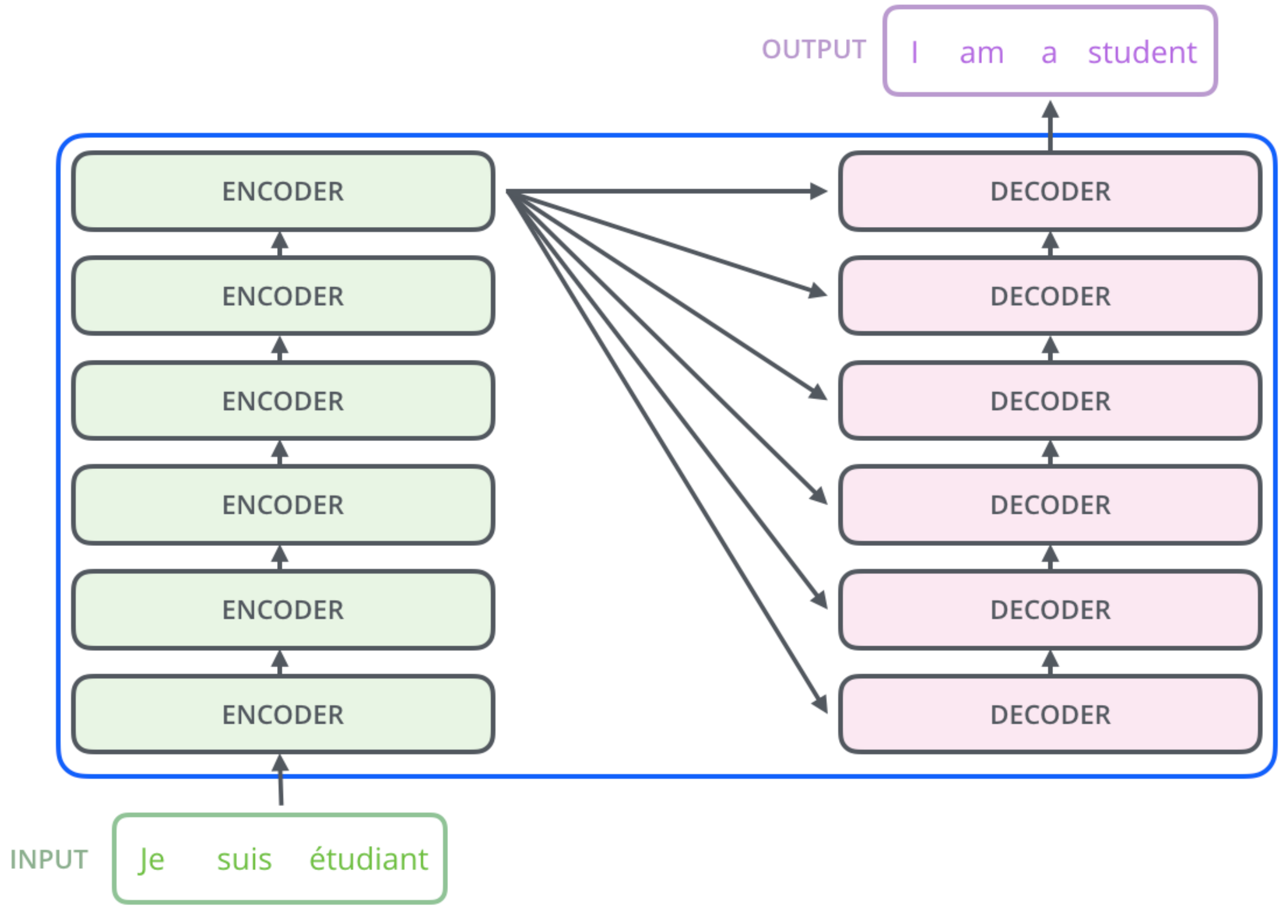
# Transformer Networks, Vision Transformers



Budapest, 11th November 2025

## Transformers [1]

- Multi-Head Attention
- Self-Attention

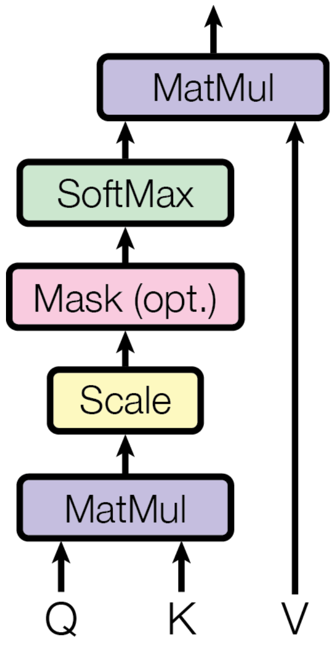


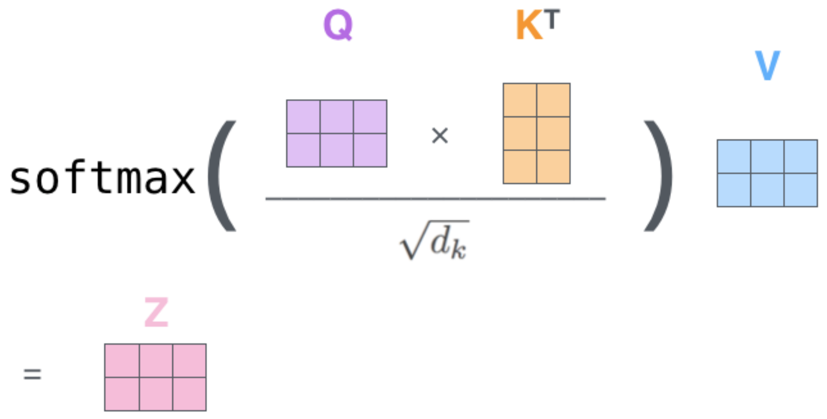
[1] Vaswani, Ashish et al. “Attention is All you Need.” (2017)

## Transformer Attention

- **Feature-based attention: The Key, Value, and Query**
- Key-value-query concepts come from **information retrieval systems**.
- Example of searching for a video on YouTube:
    - When you search (**query**) for a particular video, the search engine will map your **query** against a set of **keys** (video title, description, etc.) associated with possible stored videos. Then the algorithm will present you the best-matched videos (**values**). This is the foundation of content/**feature-based lookup**.
- Bringing this idea closer to the transformer's attention we have something like this:

## Transformer Attention





## Self-Attention

**Self-attention: the key component of the Transformer architecture**

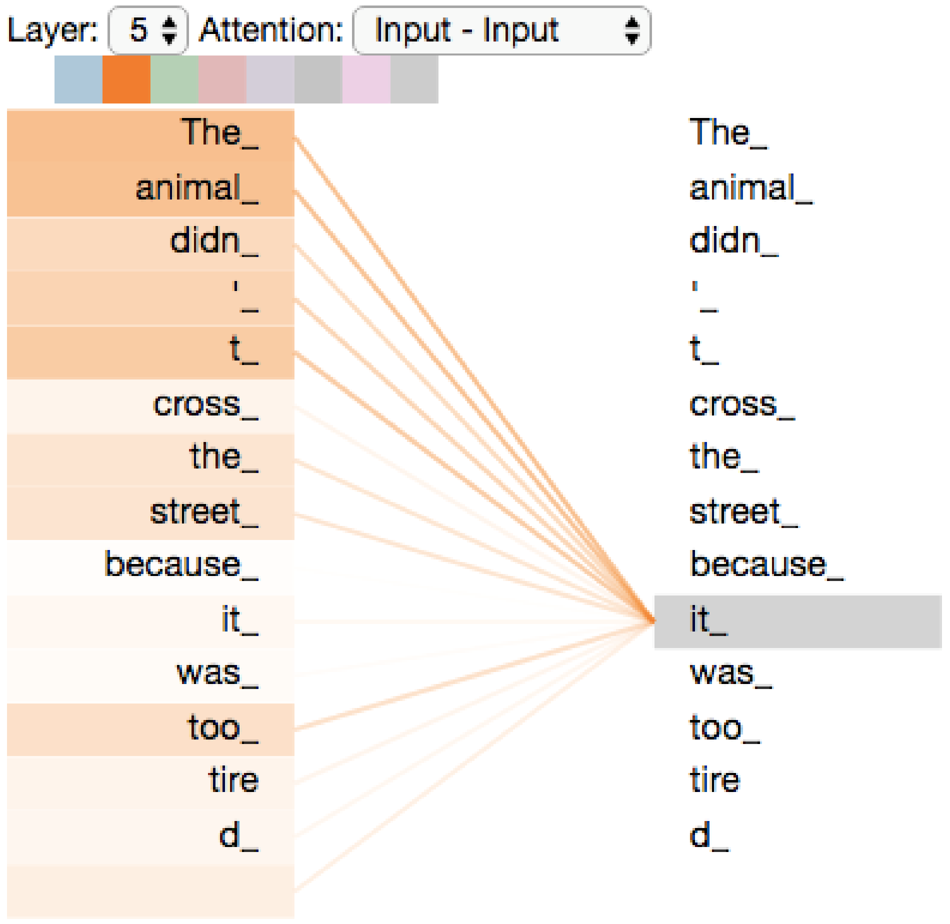
- We can also define the attention of the same sequence, called self-attention. Instead of looking for an input-output sequence association/alignment, **we are now looking for scores between the elements of the sequence**

Self-attention
Probability score matrix

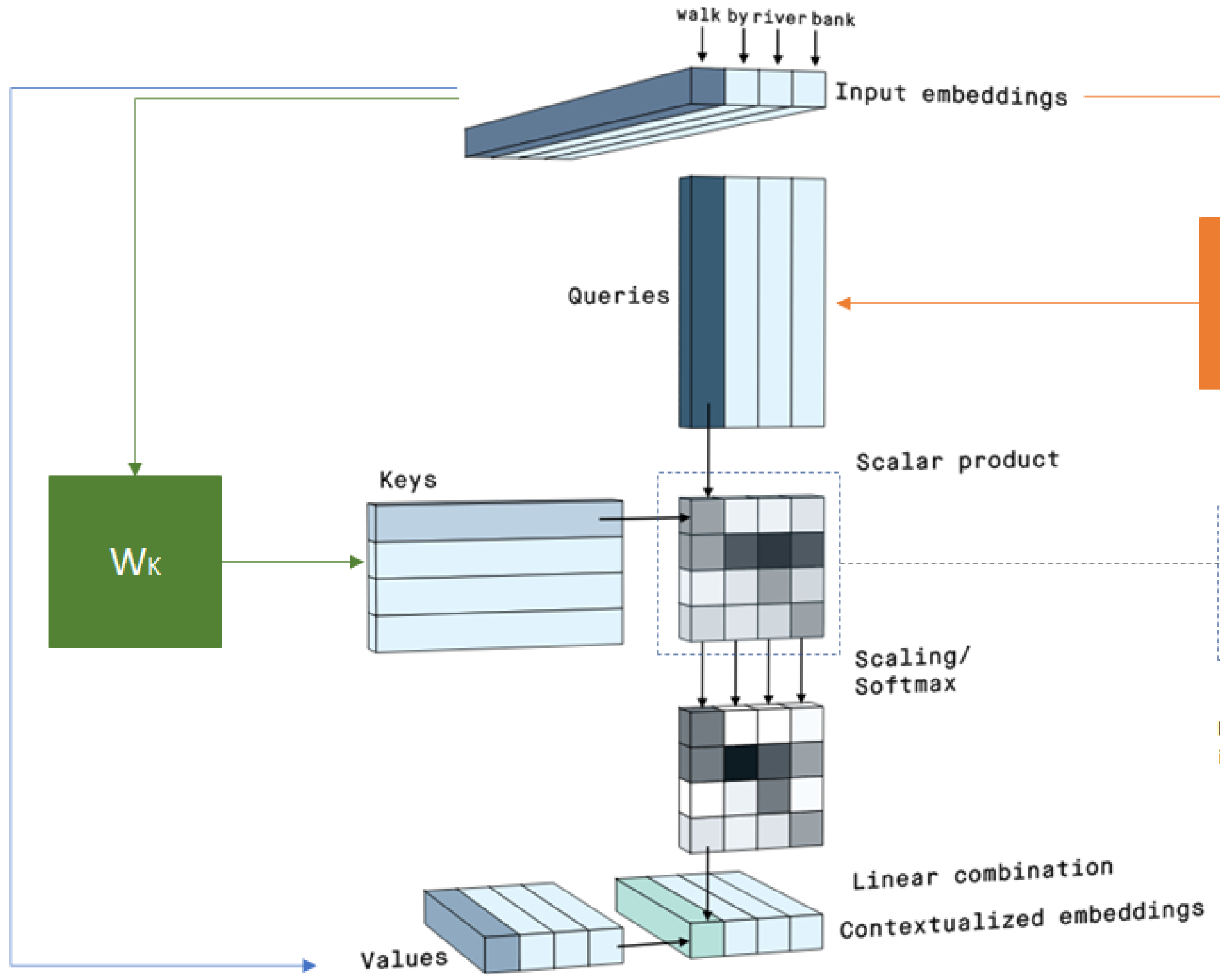
| | Hello | I | love | you |
|---|---|---|---|---|
| Hello | 0.8 | 0.1 | 0.05 | 0.05 |
| I | 0.1 | 0.6 | 0.2 | 0.1 |
| love | 0.05 | 0.2 | 0.65 | 0.1 |
| you | 0.2 | 0.1 | 0.1 | 0.6 |

Softmax(Attention)
equation

## Self-Attention



## Visualizing Self-Attention



$$\text{softmax}\left(\frac{\underset{Q}{\boxed{\phantom{x}}} \times \underset{K^T}{\boxed{\phantom{x}}}}{\sqrt{d_k}}\right) \underset{V}{\boxed{\phantom{x}}} = \underset{Z}{\boxed{\phantom{x}}}$$

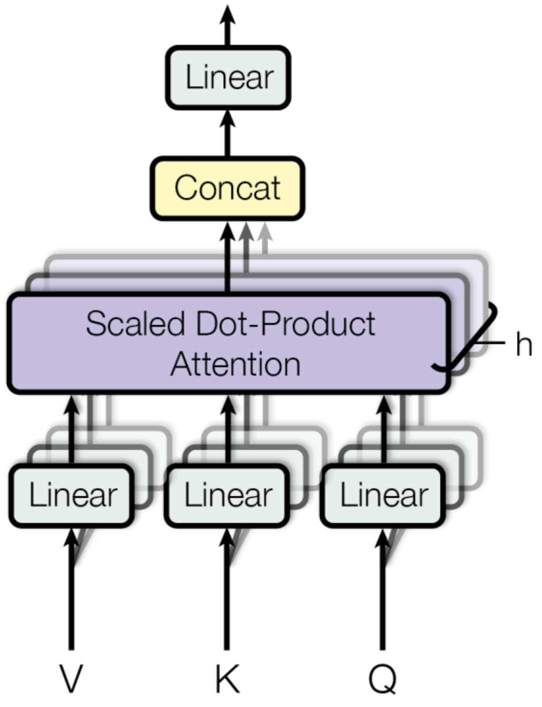
Embedded tensor in Q space

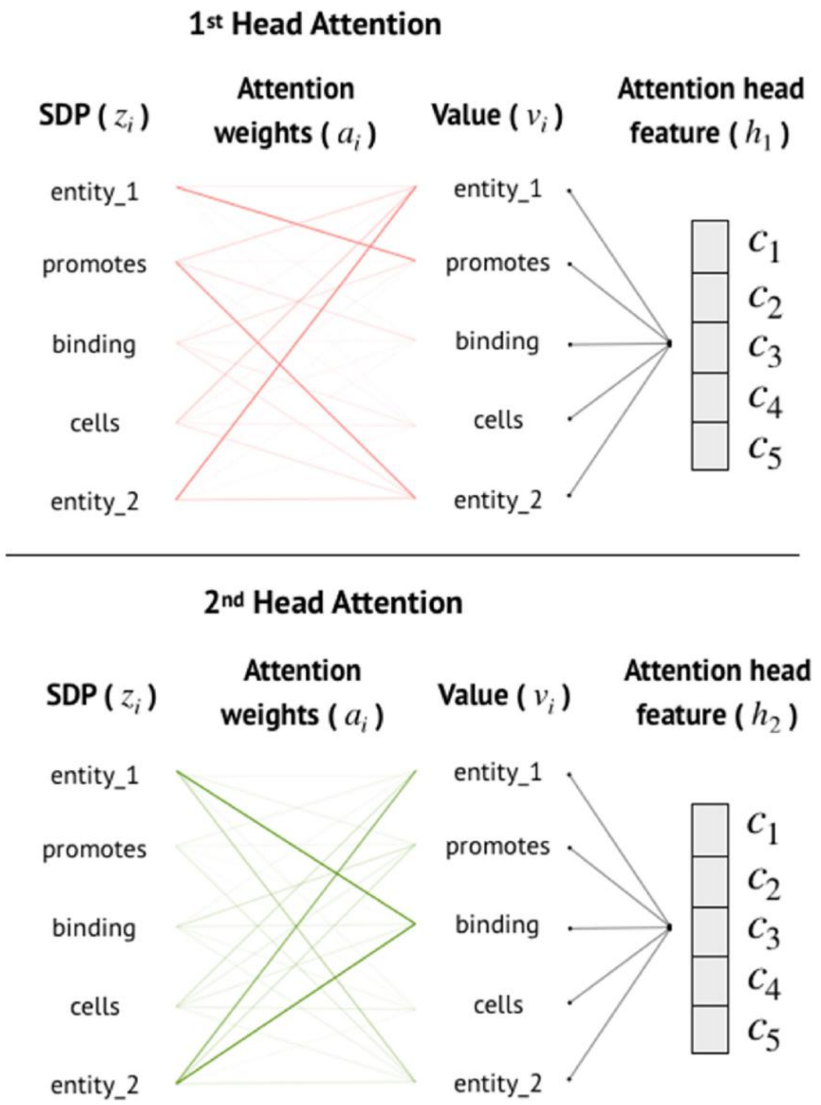
| Self Attention | walk | by | river | bank |
|---|---|---|---|---|
| walk | 0.88 | 0.1 | 0.21 | 0.12 |
| by | 0.08 | 0.8 | 0.12 | 0.14 |
| river | 0.03 | 0.14 | 0.83 | 0.7 |
| bank | 0.01 | 0.01 | 0.4 | 0.77 |

Embedded tensor in K space

## Multi-Head Attention

- Repeating self-attention multiple times to better understand the input (multiple ways of deciding the relationship between the parts of the input)
    - Looks at the input from different perspectives

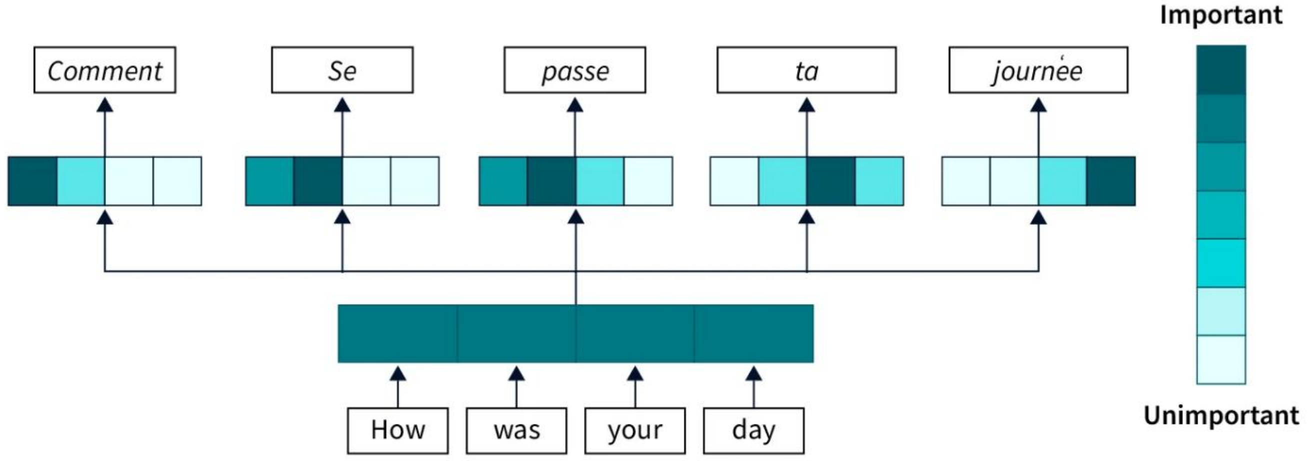




## Summary

### 1. Attention

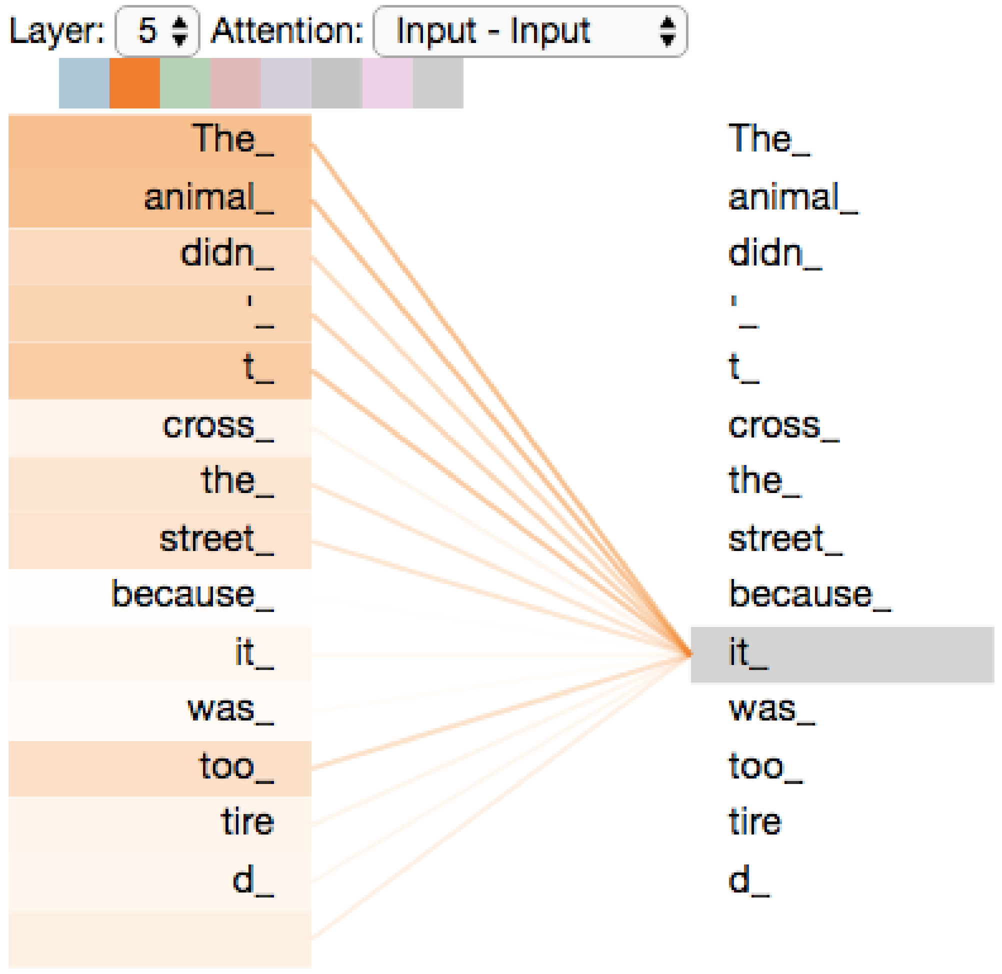
- How does the input elements relate to the output (what part of the input is relevant for the output)
    - Helps the model in focusing on the most relevant information



## Summary

### 2. Self-Attention

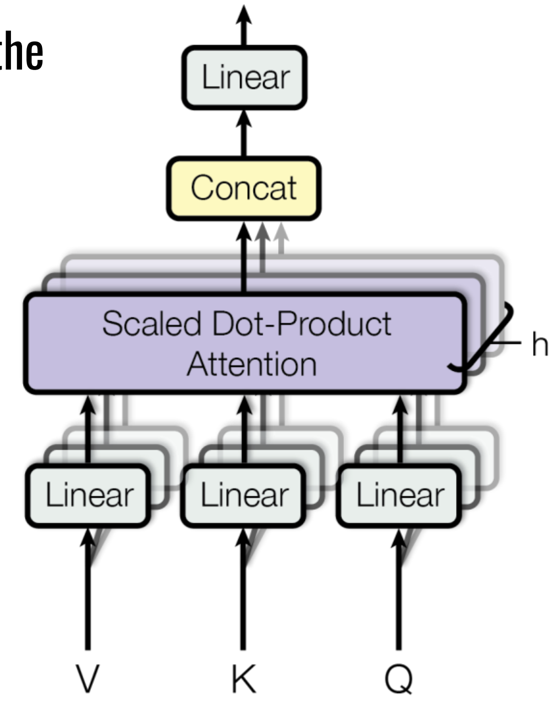
- How do the different elements of the input relate to each other (what part of the input is relevant in understanding the other parts of the input)
    - Helps in capturing dependencies and contextual information within the input

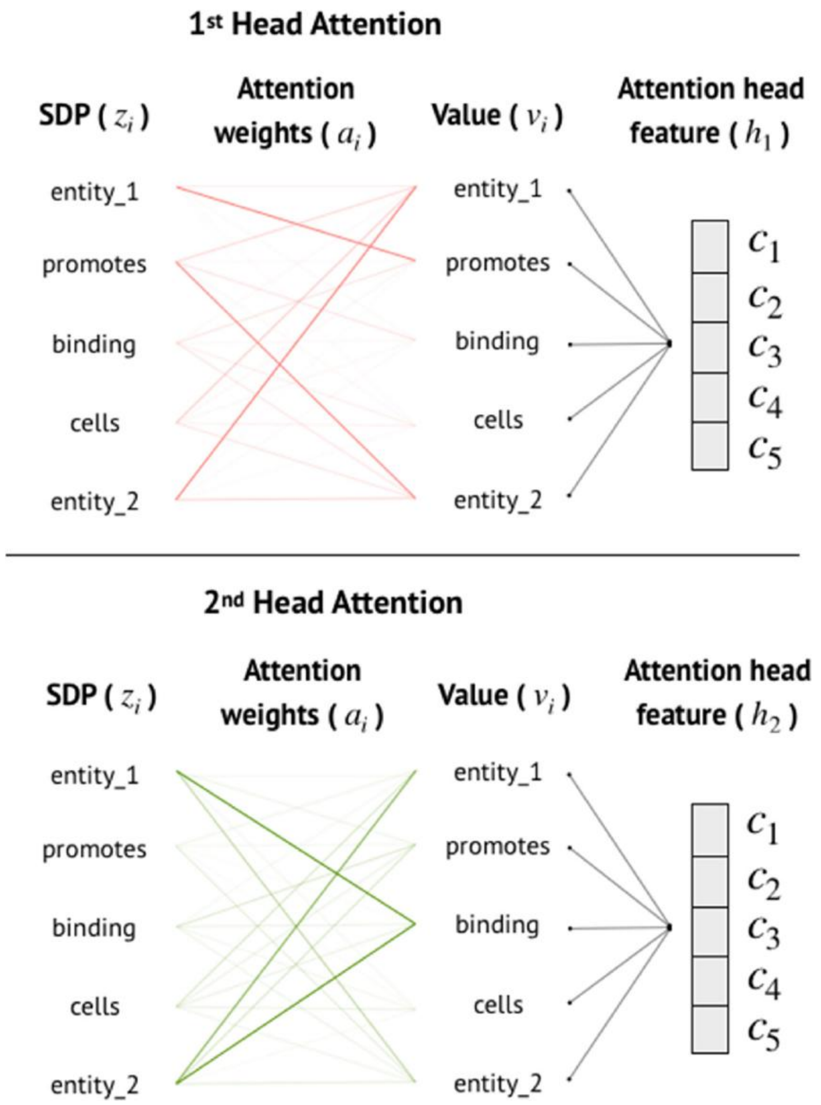


## Summary

### 3. Multi-Head Attention

- Repeating self-attention multiple times to better understand the input (multiple ways of deciding the relationship between the parts of the input)
  - Looks at the input from different perspectives

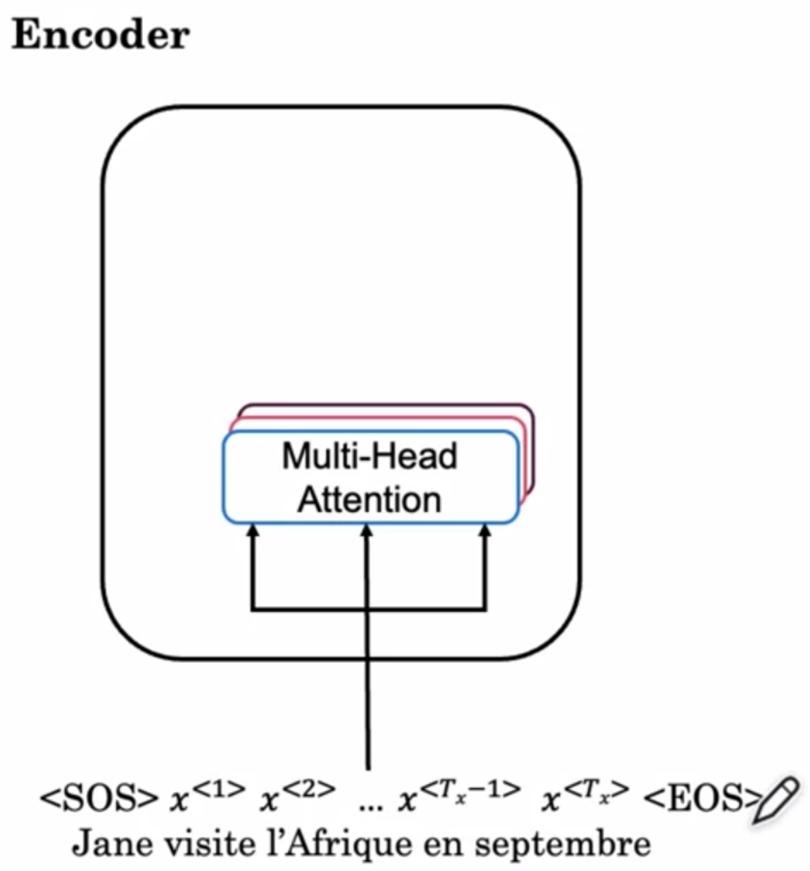




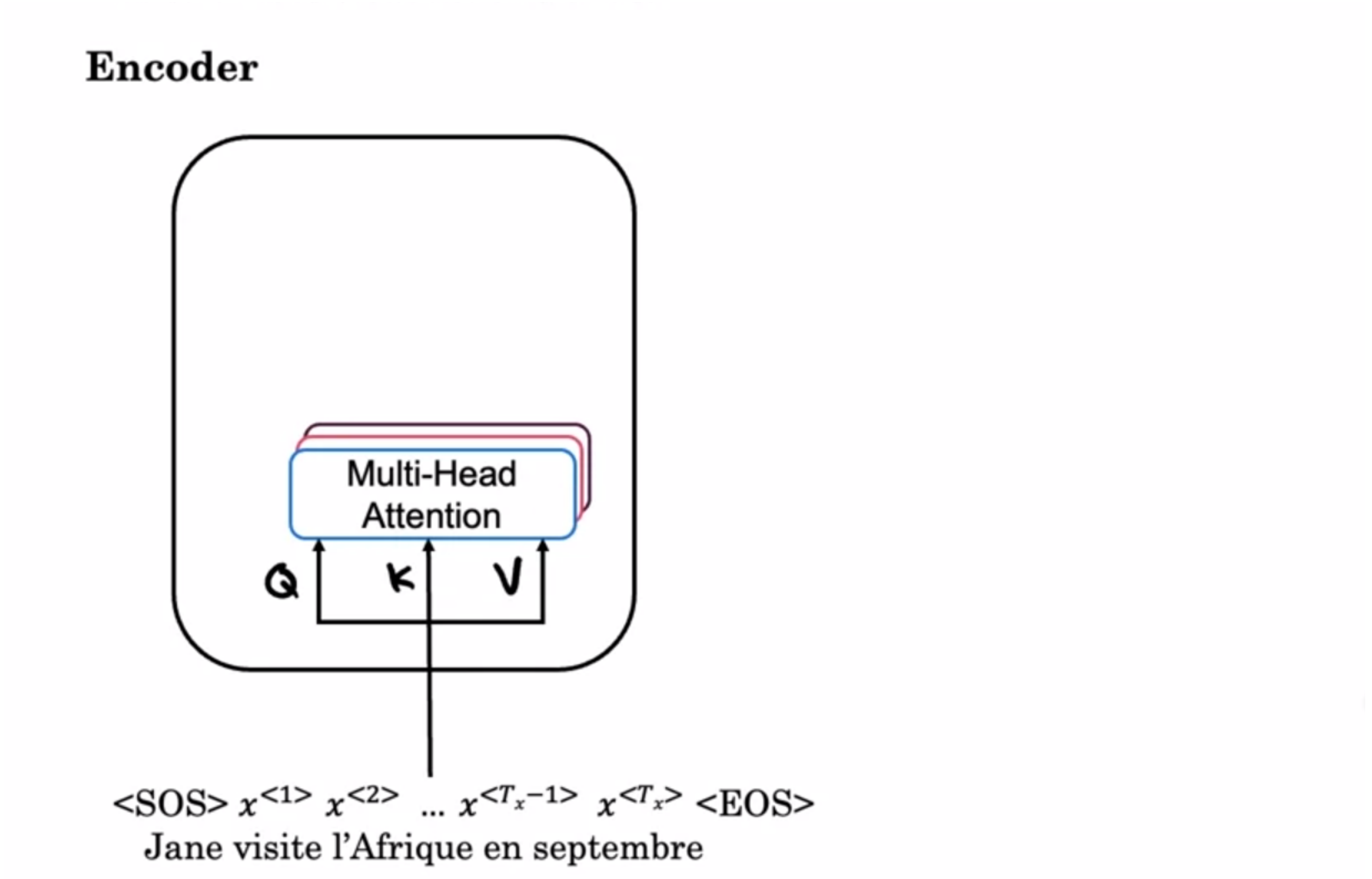
## Transformer

$\text{<SOS>}\ x^{<1>}\ x^{<2>}\ \ldots\ x^{<T_x-1>}\ x^{<T_x>}\ \text{<EOS>}$
Jane visite l'Afrique en septembre

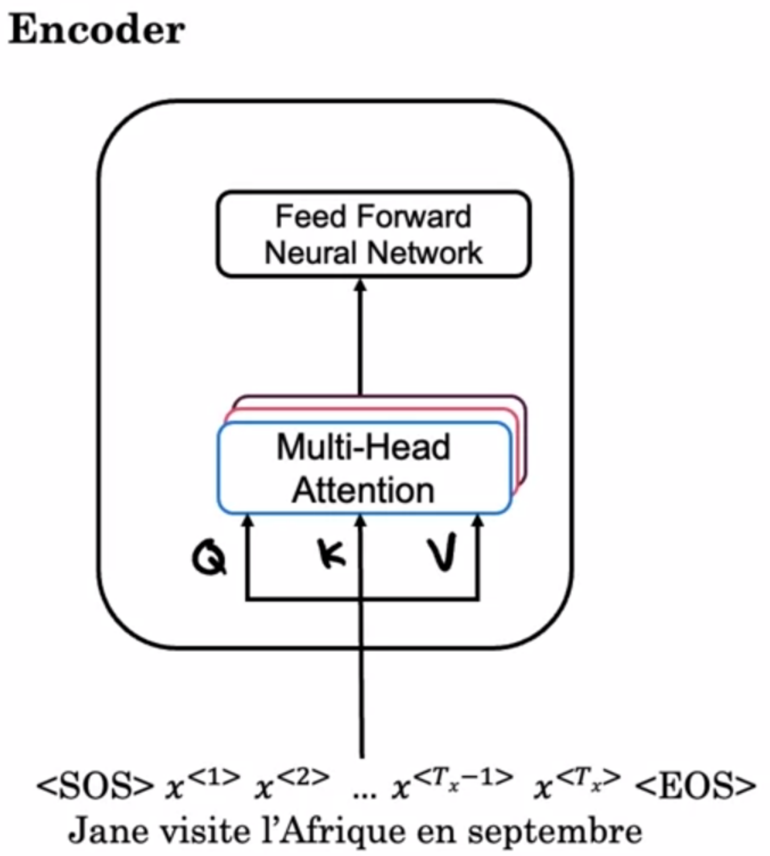
## Transformer


Encoder
Multi-Head Attention
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

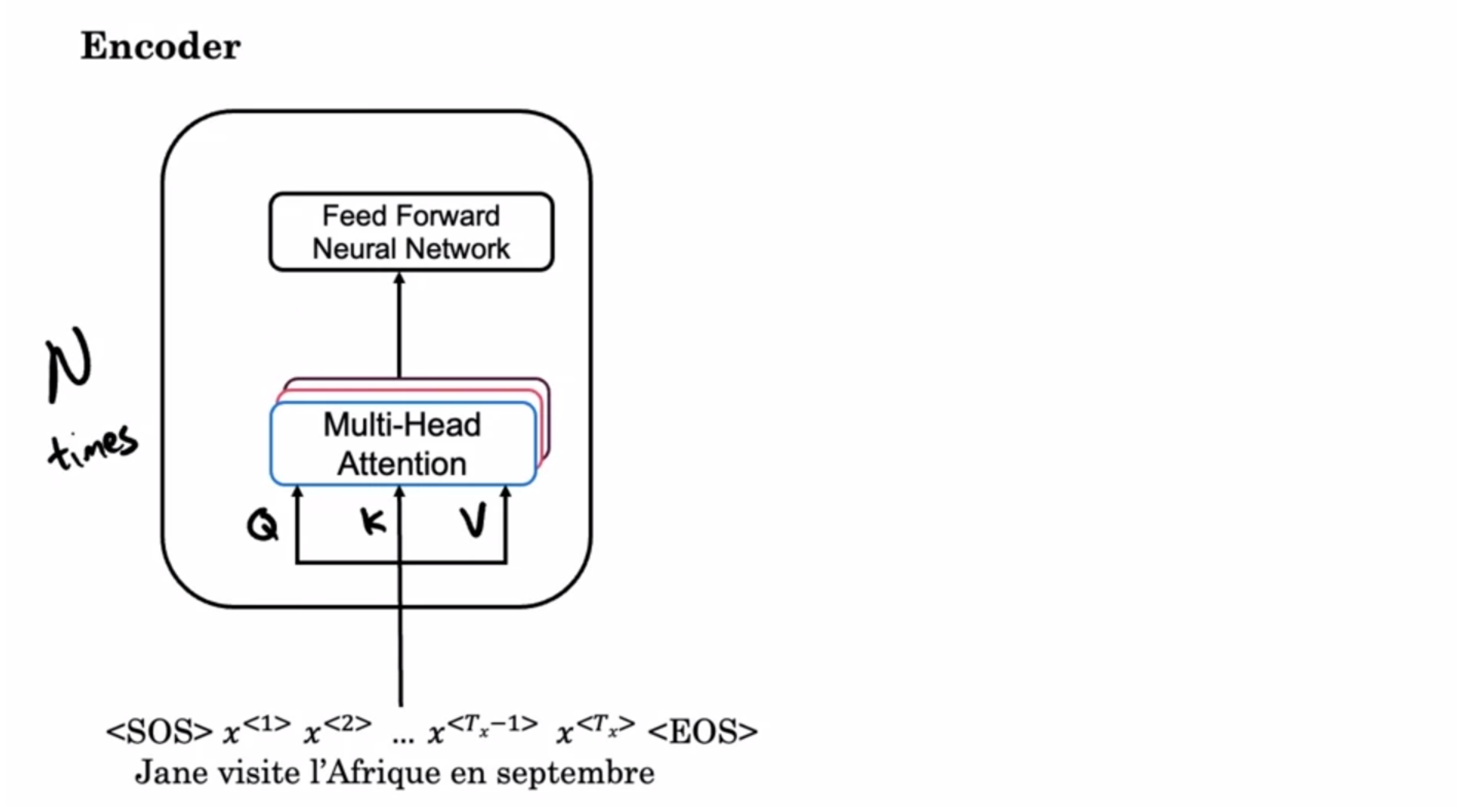
## Transformer


Encoder
Multi-Head Attention
Q
K
V
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

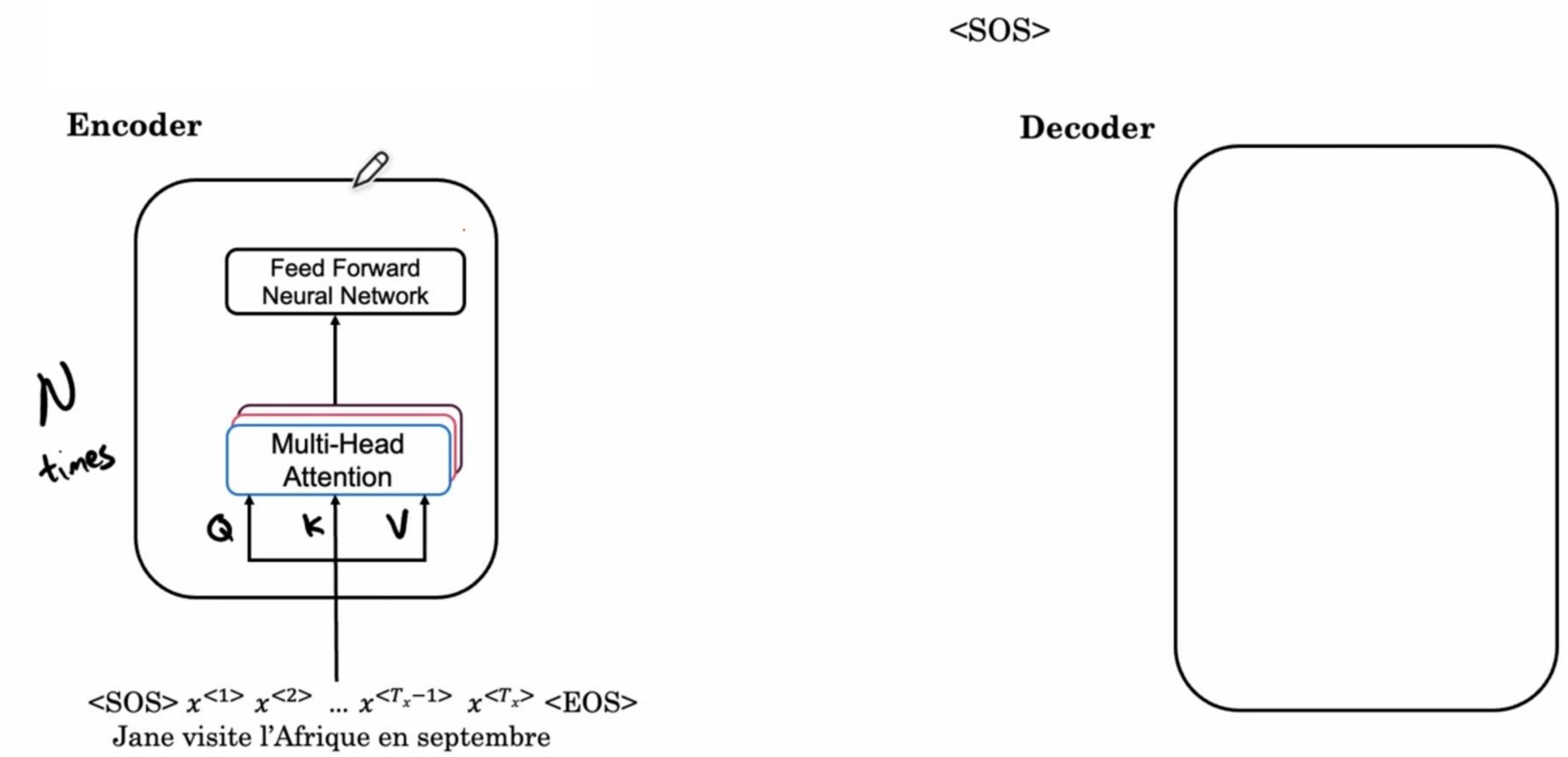
## Transformer


Encoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

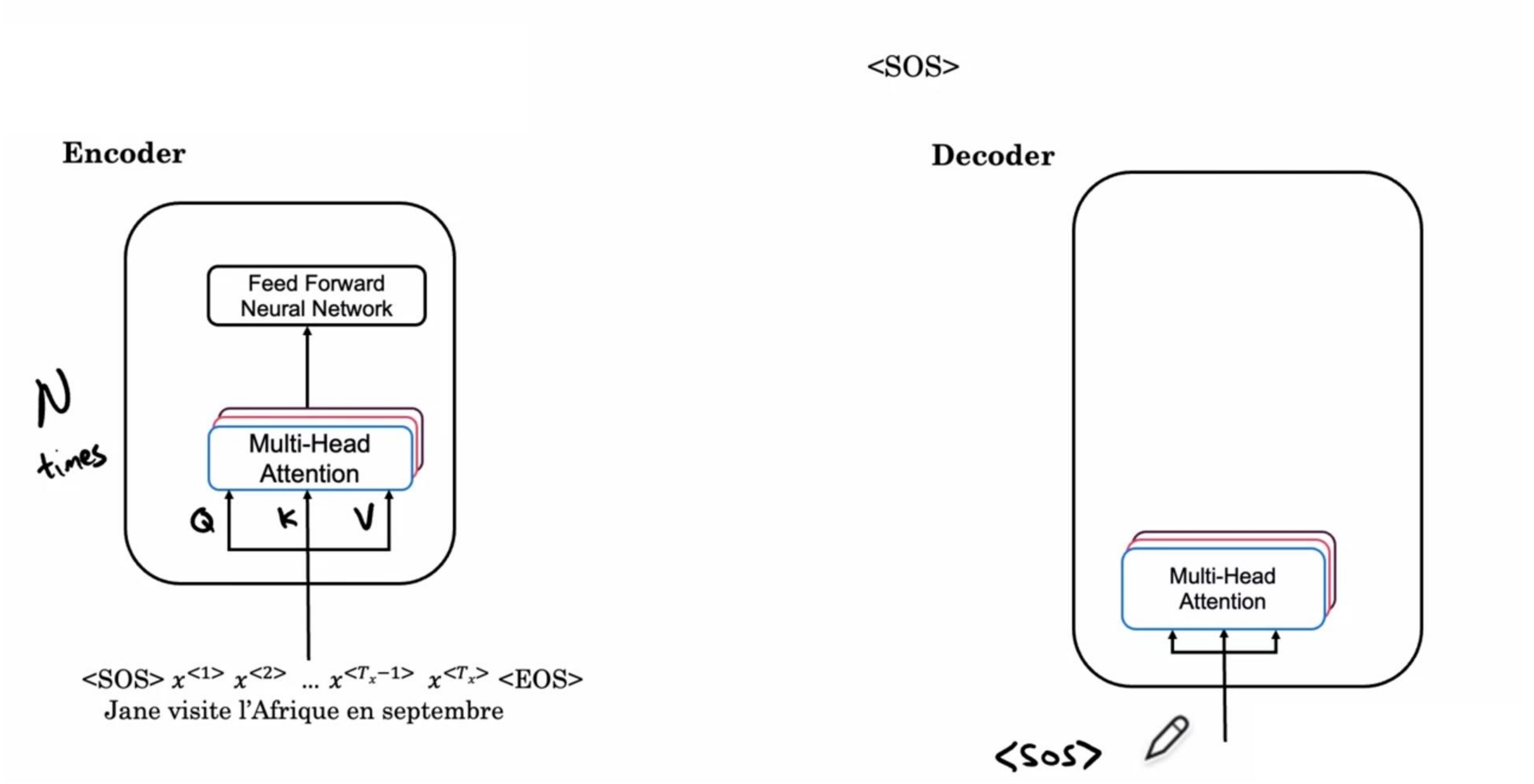
## Transformer


Encoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

## Transformer

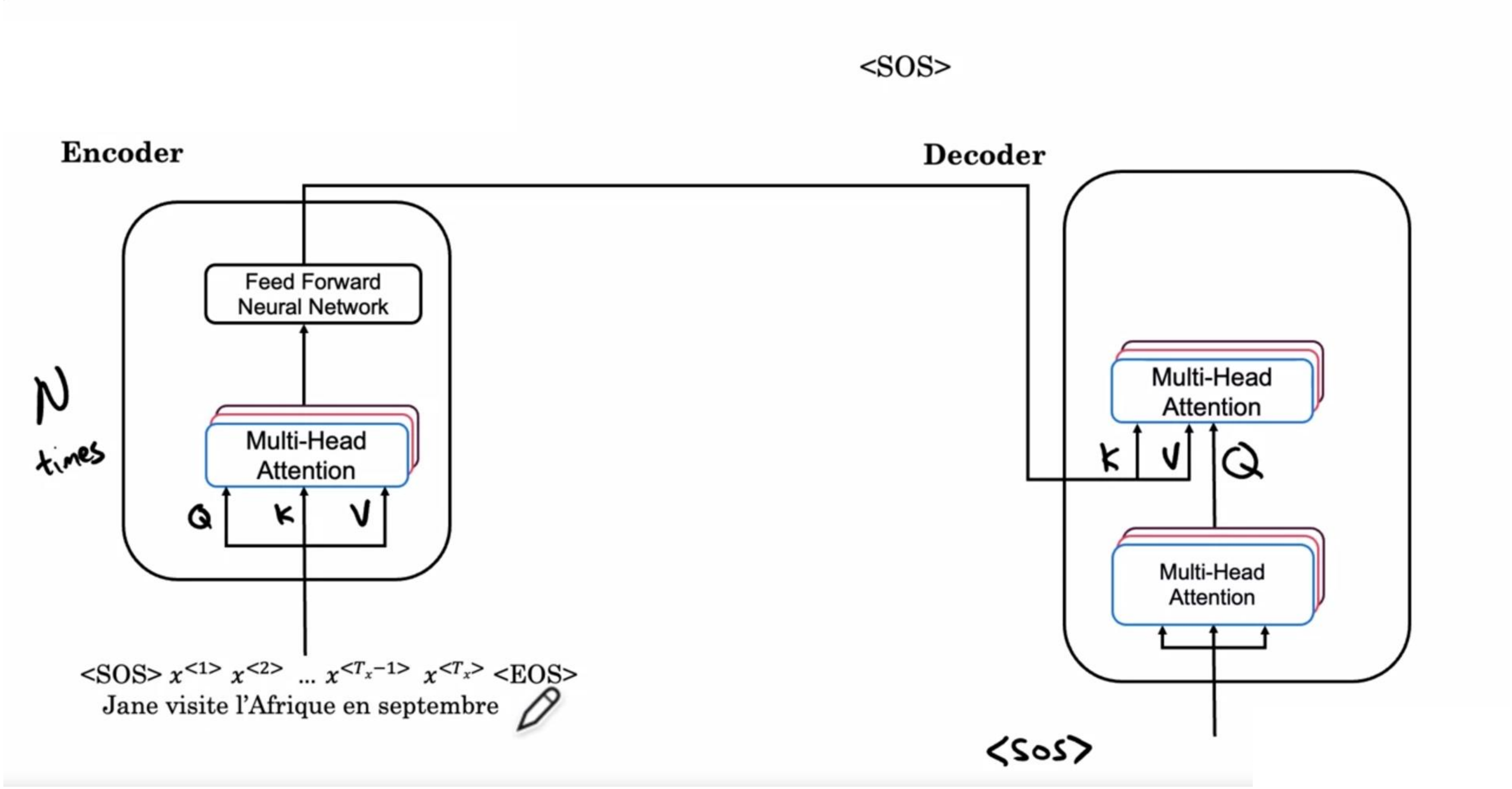

<SOS>
Encoder
Decoder
Feed Forward Neural Network
N times
Multi-Head Attention
Q
K
V
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

## Transformer

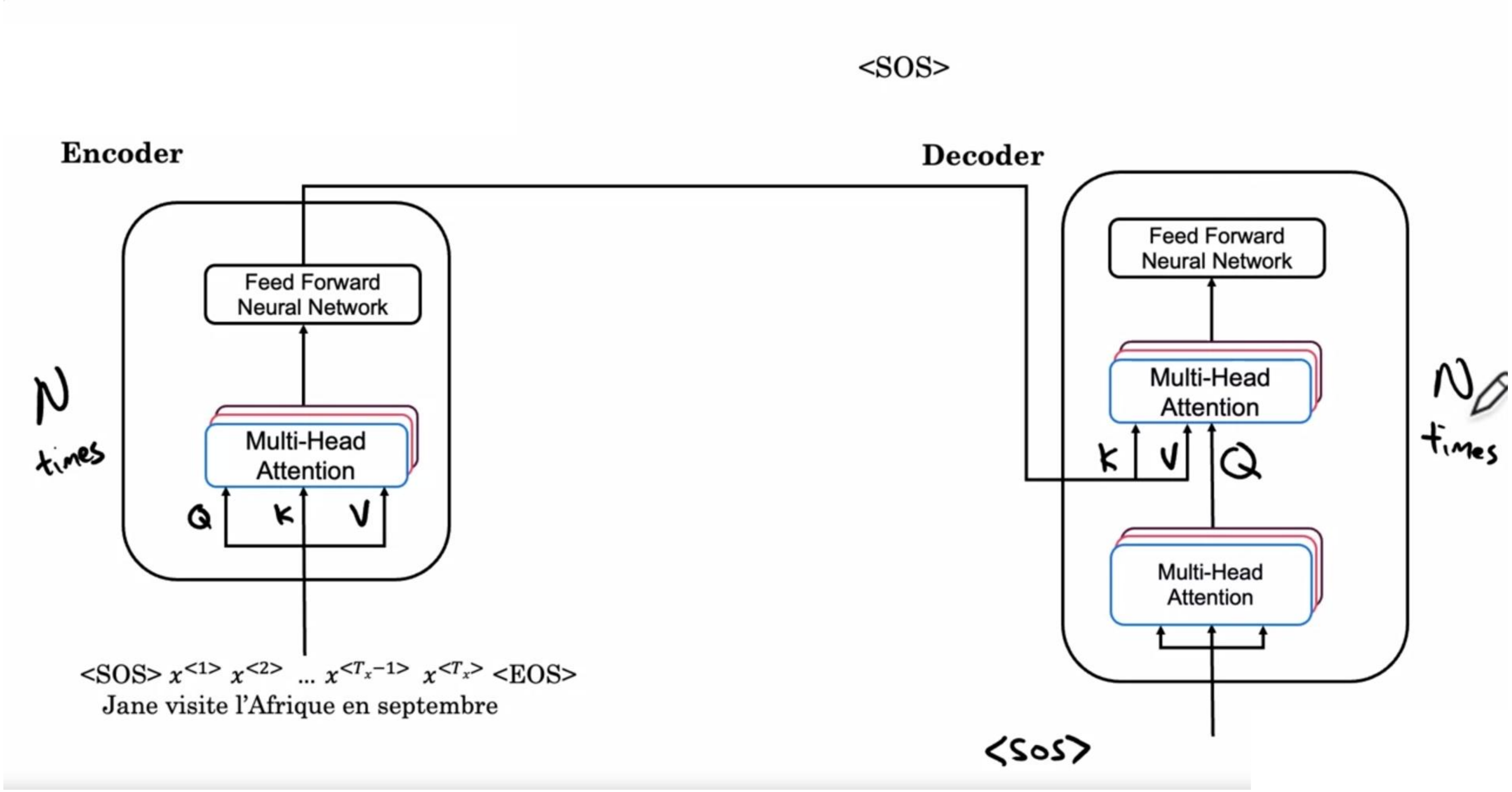


<SOS>
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
$\langle SOS \rangle\ x^{<1>}\ x^{<2>}\ \dots\ x^{<T_x-1>}\ x^{<T_x>}\ \langle EOS \rangle$
Jane visite l'Afrique en septembre
Multi-Head Attention
<SOS>

## Transformer

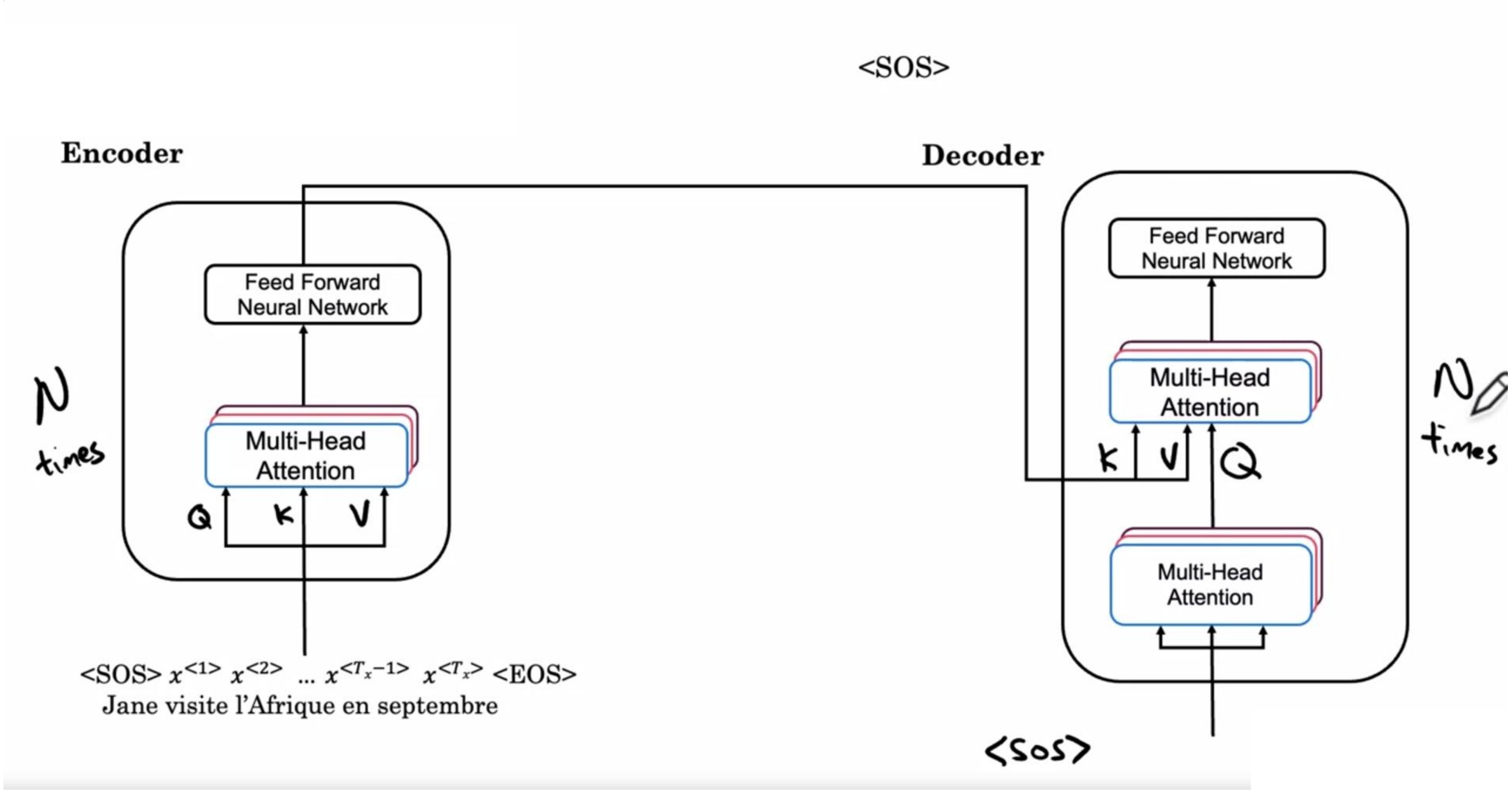

<SOS>
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
N times
Q
K
V
Multi-Head Attention
K
V
Q
Multi-Head Attention
$<SOS>\ x^{<1>}\ x^{<2>}\ \dots\ x^{<T_x-1>}\ x^{<T_x>}\ <EOS>$
Jane visite l'Afrique en septembre
<SOS>

## Transformer



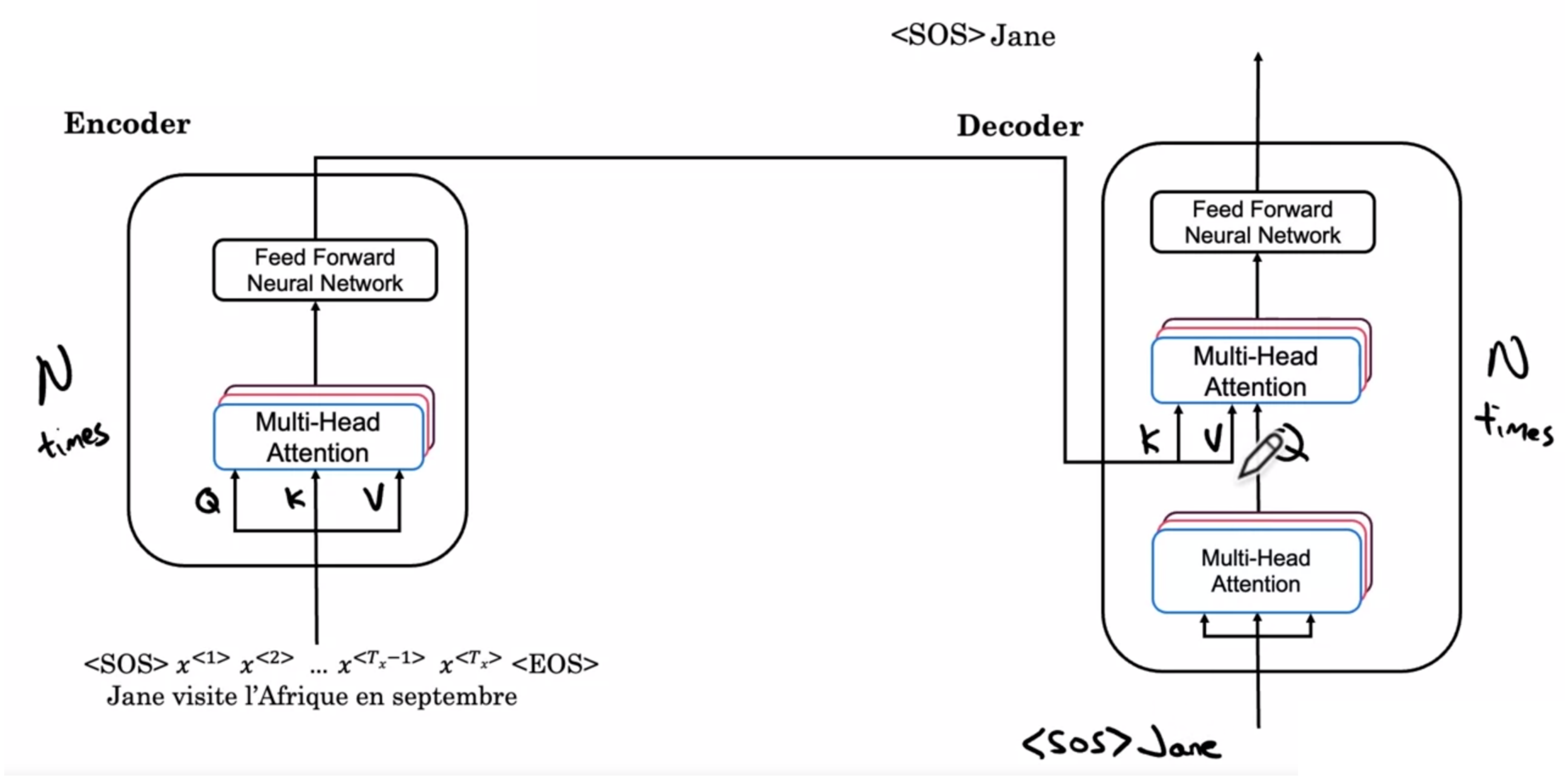
<SOS>
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS>

## Transformer

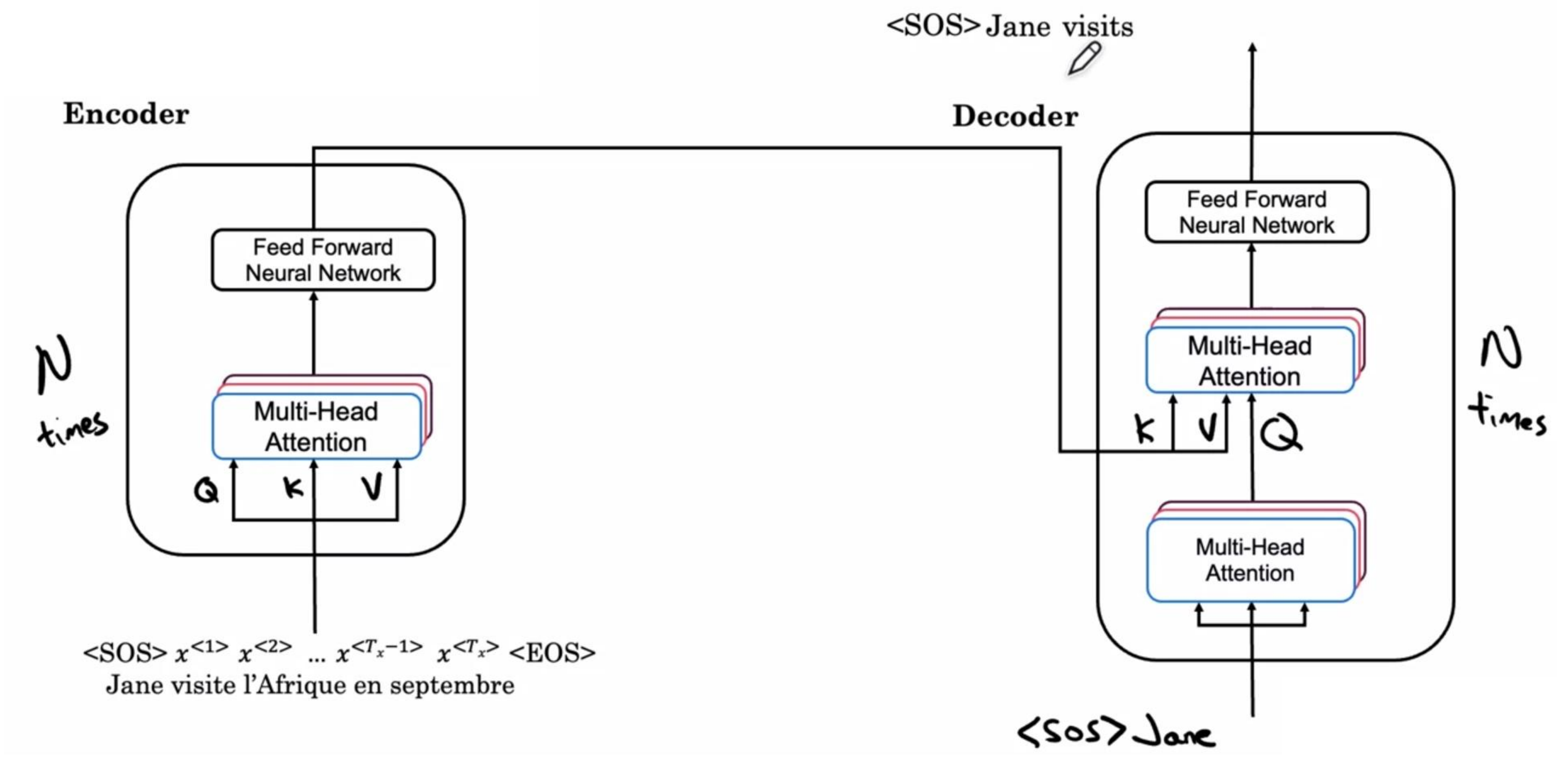


<SOS>
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS>

## Transformer

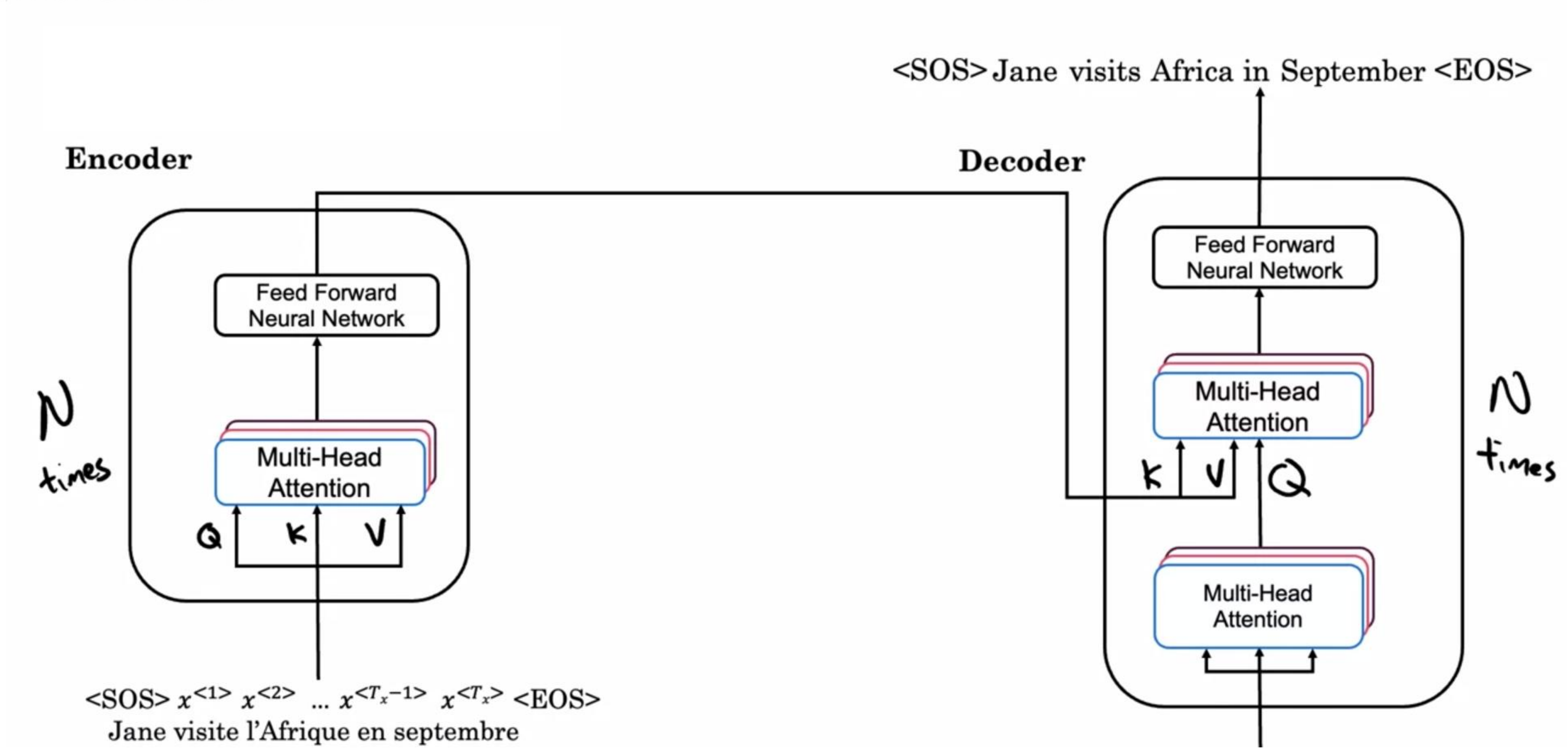

<SOS> Jane
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS> Jane

## Transformer



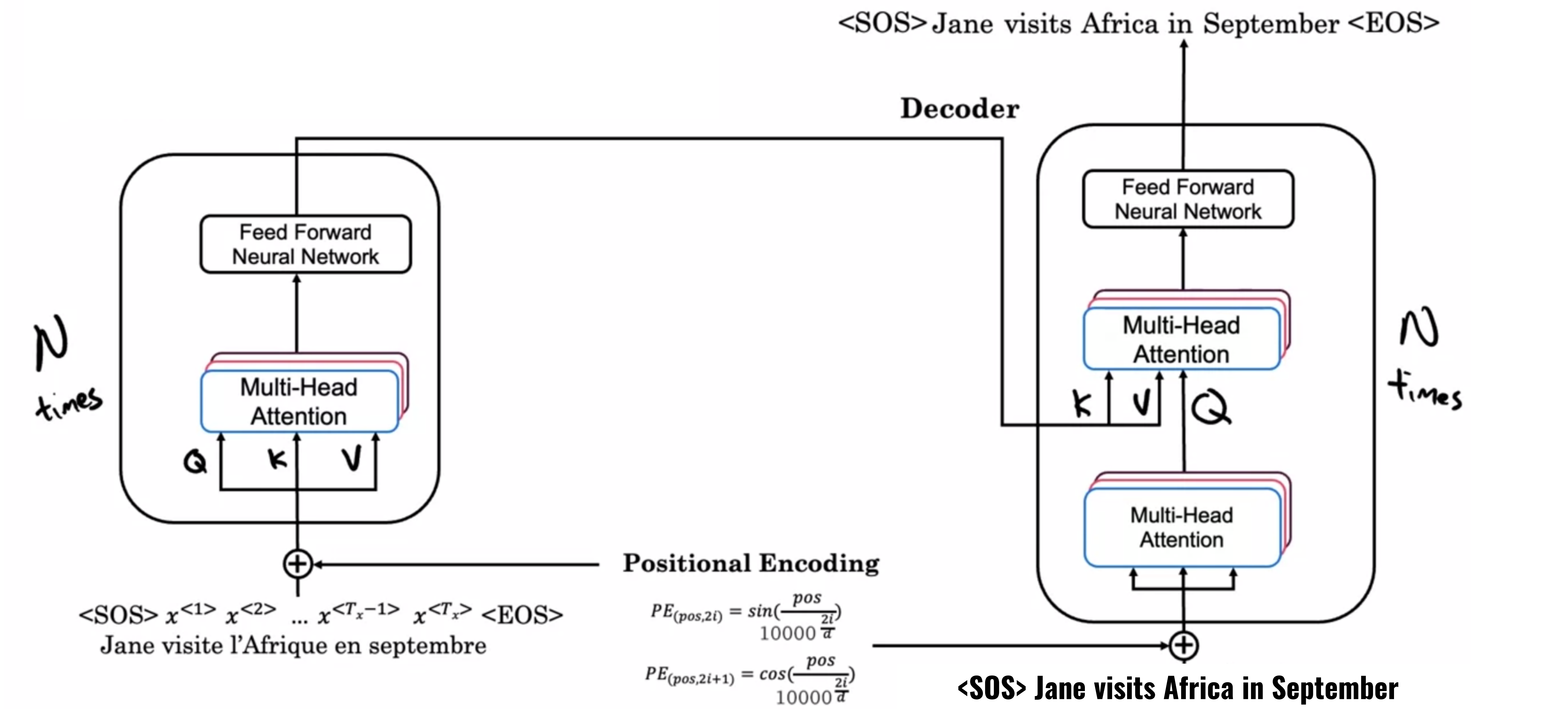
<SOS> Jane visits
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS> Jane

## Transformer



<SOS> Jane visits Africa in September <EOS>
Encoder
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
$<SOS>\ x^{<1>}\ x^{<2>}\ \dots\ x^{<T_x-1>}\ x^{<T_x>}\ <EOS>$
Jane visite l'Afrique en septembre

## Transformer

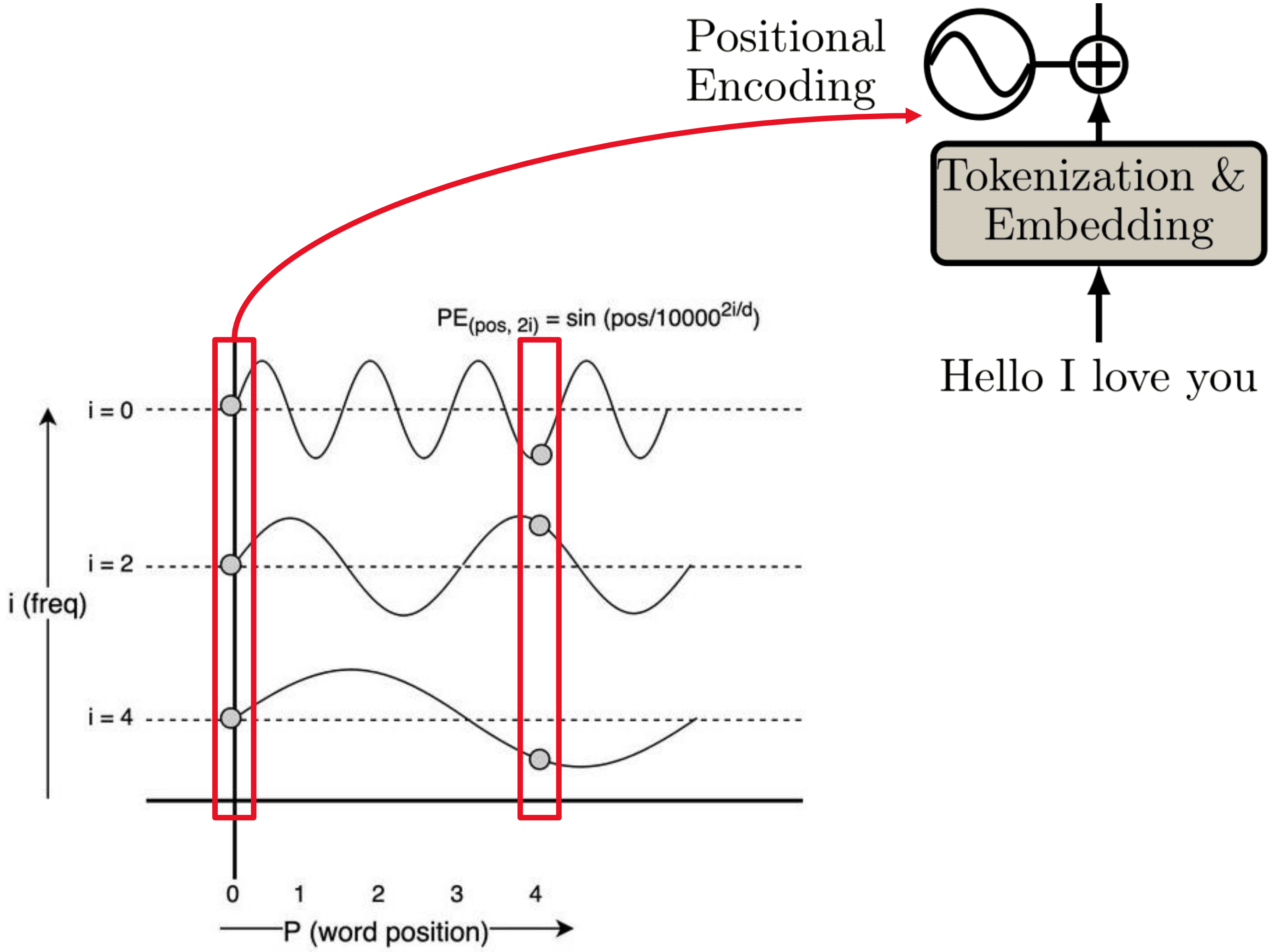


<SOS> Jane visits Africa in September <EOS>
Decoder
Feed Forward Neural Network
Multi-Head Attention
Q
K
V
N times
Feed Forward Neural Network
Multi-Head Attention
K
V
Q
Multi-Head Attention
N times
Positional Encoding
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d}}})$
$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d}}})$

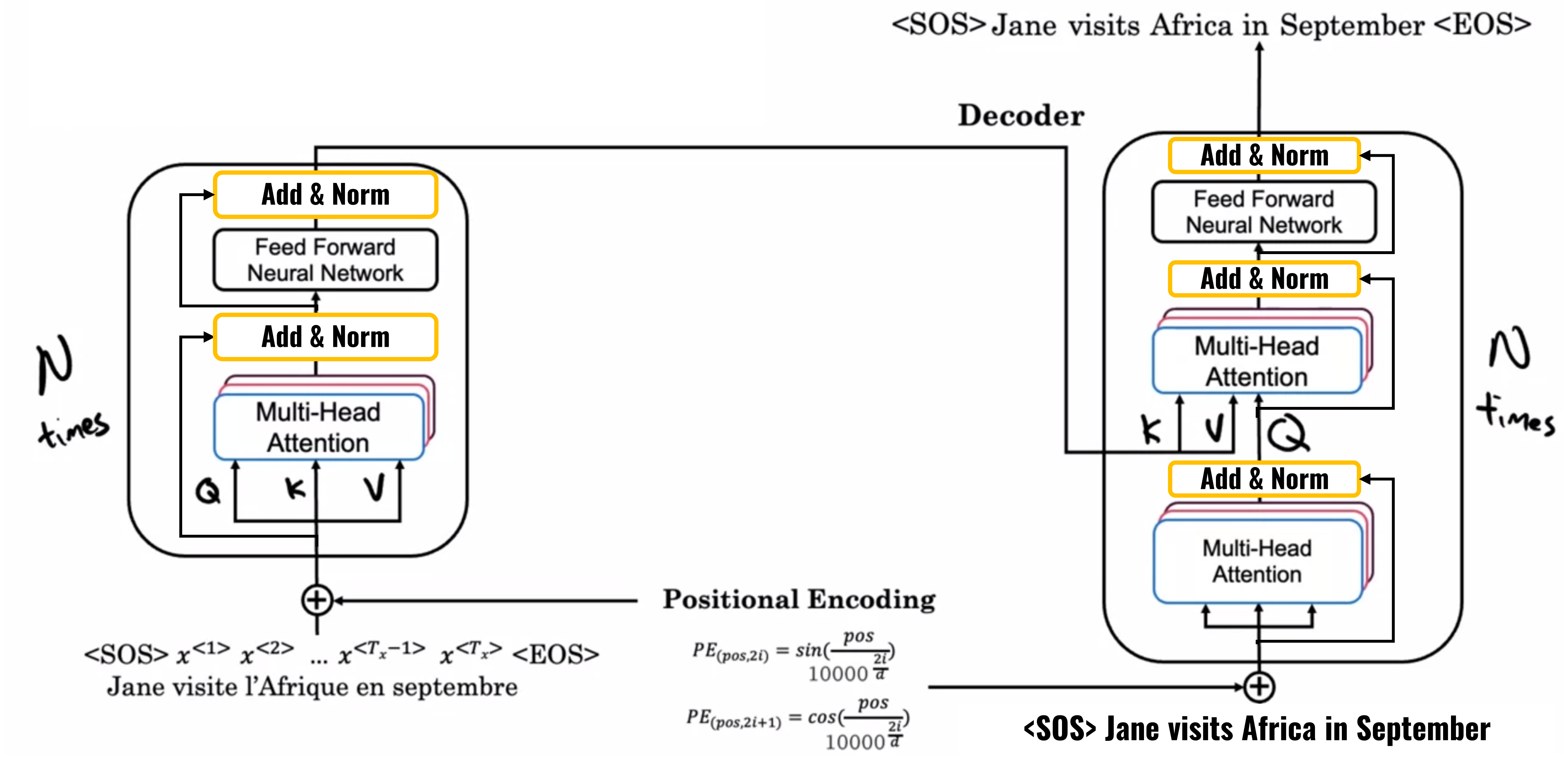
**<SOS> Jane visits Africa in September**

## Transformer

- When you convert a sequence into a set (**tokenization**), you lose the notion of order.
- **Positional encoding** is a set of small constants, which are added to the word embedding vector before the first self-attention layer.
- So if the same word appears in a different position, the actual representation will be slightly different, **depending on where it appears in the input sentence**.
- They use a sinusoidal function for the positional encoding. The sine function tells the model to pay attention to a particular wavelength.



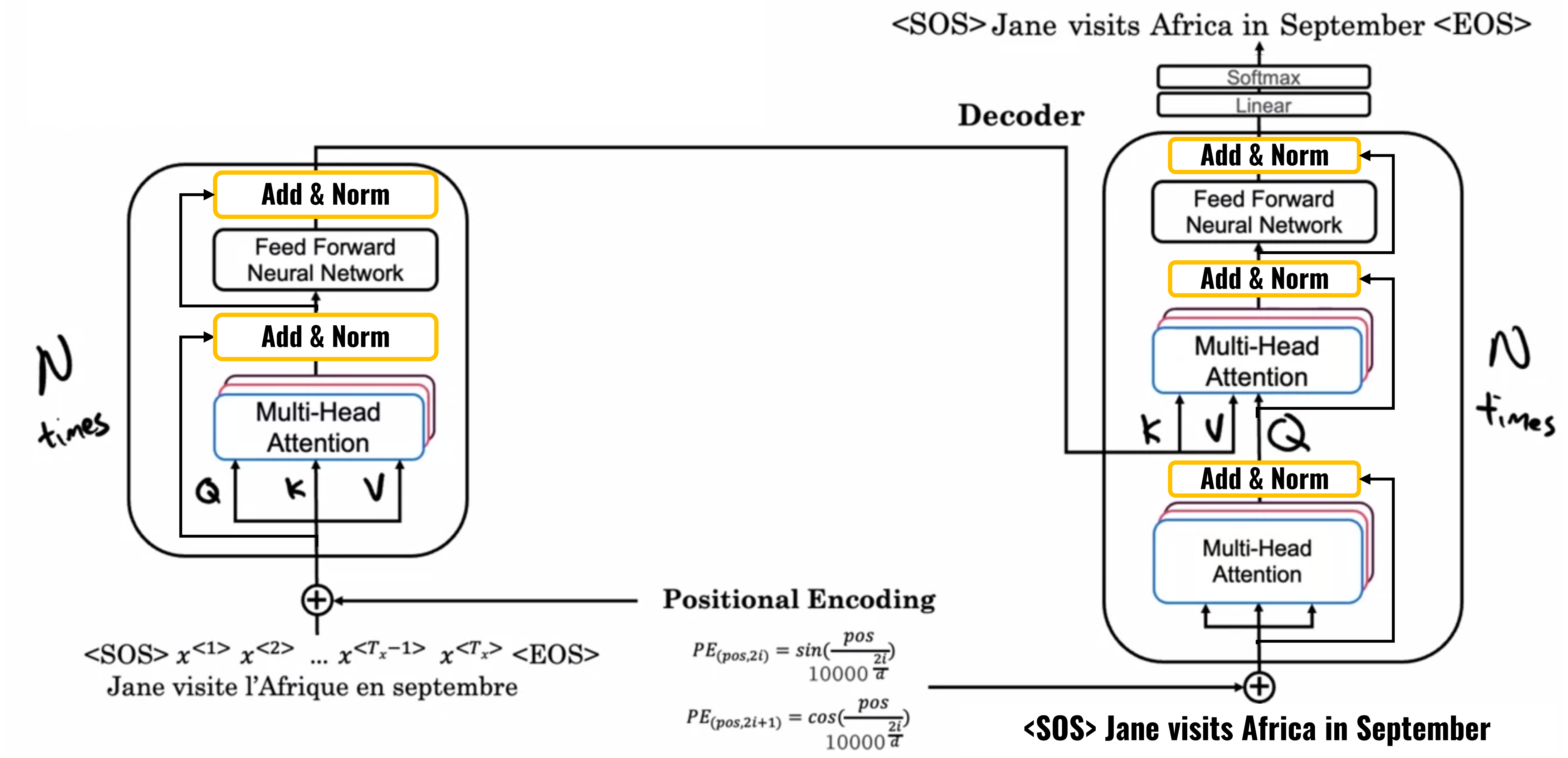
## Transformer


<SOS> Jane visits Africa in September <EOS>
Decoder
Add & Norm
Feed Forward Neural Network
Add & Norm
Multi-Head Attention
Q
K
V
N times
K
V
Q
Add & Norm
Multi-Head Attention
N times
Positional Encoding
<SOS> Jane visits Africa in September
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d}}})$$

## Transformer

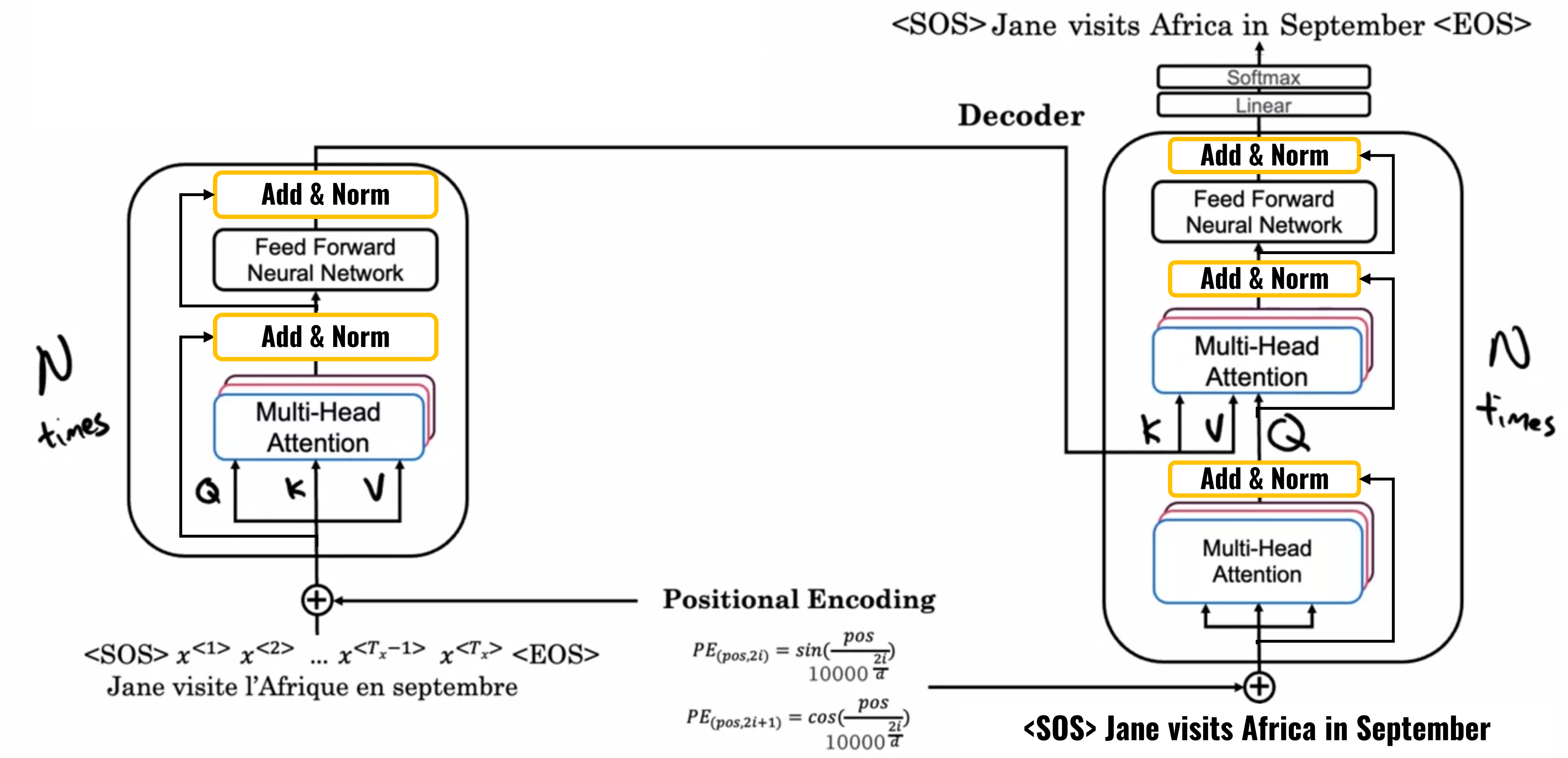


<SOS> Jane visits Africa in September <EOS>
Softmax
Linear
Decoder
Add & Norm
Feed Forward Neural Network
Add & Norm
Multi-Head Attention
K
V
Q
Add & Norm
Multi-Head Attention
N times
Add & Norm
Feed Forward Neural Network
Add & Norm
Multi-Head Attention
Q
K
V
N times
Positional Encoding
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS> Jane visits Africa in September

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d}}})$$

## Transformer


<SOS> Jane visits Africa in September <EOS>
Softmax
Linear
Decoder
Add & Norm
Feed Forward Neural Network
Add & Norm
Multi-Head Attention
Q
K
V
Add & Norm
Feed Forward Neural Network
Add & Norm
Multi-Head Attention
K
V
Q
Add & Norm
Multi-Head Attention
N times
N times
Positional Encoding
<SOS> $x^{<1>}$ $x^{<2>}$ ... $x^{<T_x-1>}$ $x^{<T_x>}$ <EOS>
Jane visite l'Afrique en septembre
<SOS> Jane visits Africa in September

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d}}})$$

## Transformer Encoder

**Summary: the Transformer encoder**

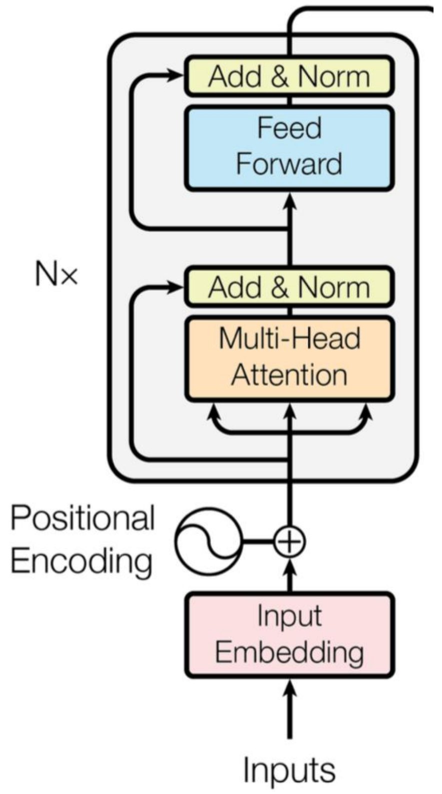
To process a sentence, we need these 3 steps:

1. Word embeddings of the input sentence are computed simultaneously.
2. Positional encodings are then applied to each embedding resulting in word vectors that also include positional information.
3. The word vectors are passed to the first encoder block.

Each block consists of the following layers in the same order:

1. A multi-head self-attention layer to find correlations between each word
2. A normalization layer
3. A residual connection around the previous two sublayers
4. A linear layer
5. A second normalization layer
6. A second residual connection

Note that the above block can be replicated several times to form the Encoder. In the original paper, the encoder composed of 6 identical blocks.



# Transformer Decoder

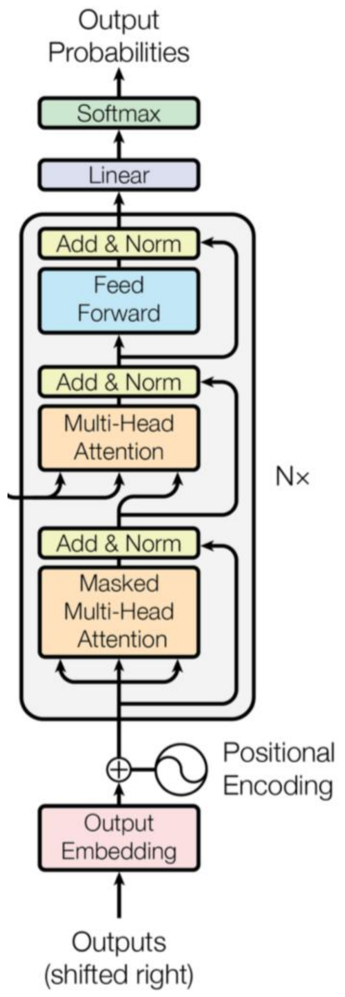
**Transformer decoder: what is different?**

The decoder consists of all the previously mentioned components plus two novel ones. As before:

1. The output sequence is fed in its entirety and word embeddings are computed
2. Positional encoding are again applied
3. And the vectors are passed to the first Decoder block

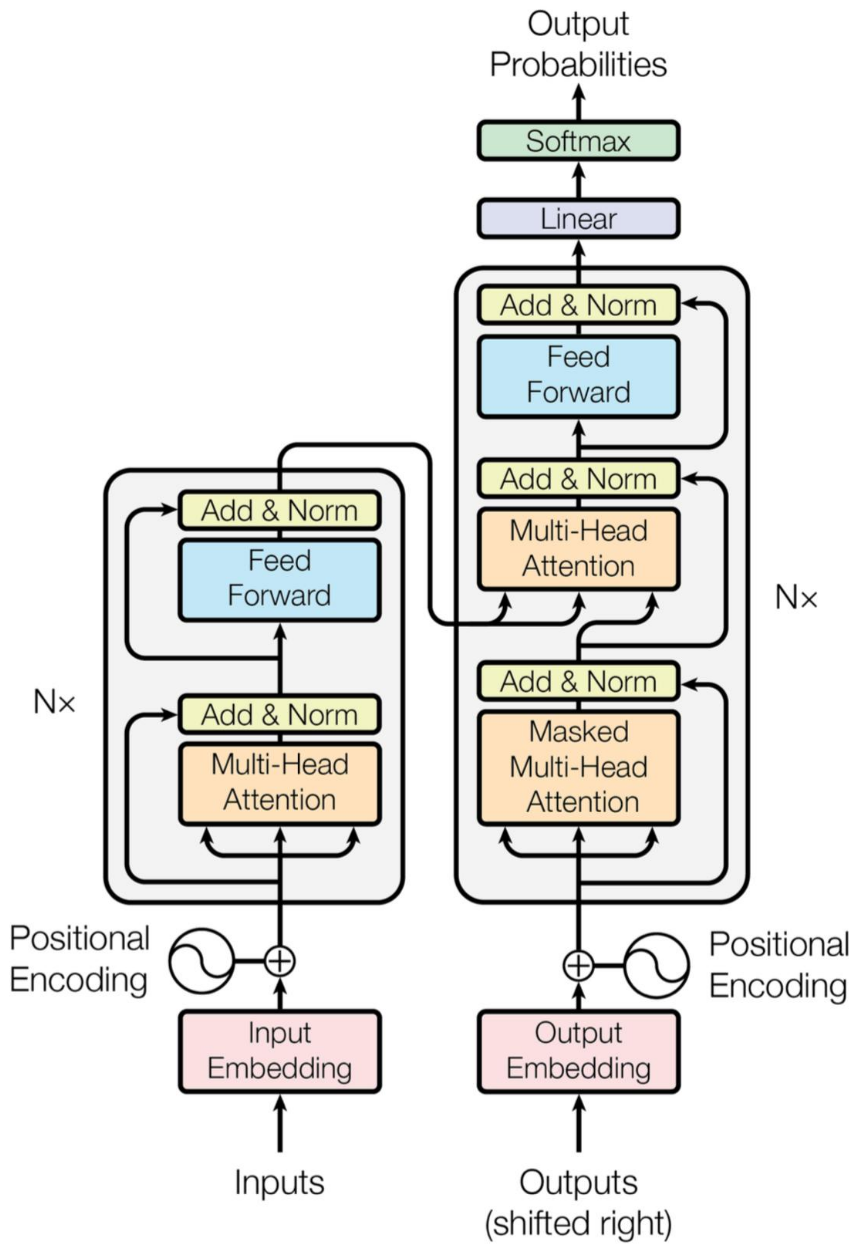
Each decoder block includes:

1. A **Masked** multi-head self-attention layer
2. A normalization layer followed by a residual connection
3. A new multi-head attention layer (known as **Encoder-Decoder attention**)
4. A second normalization layer and a residual connection
5. A linear layer and a third residual connection



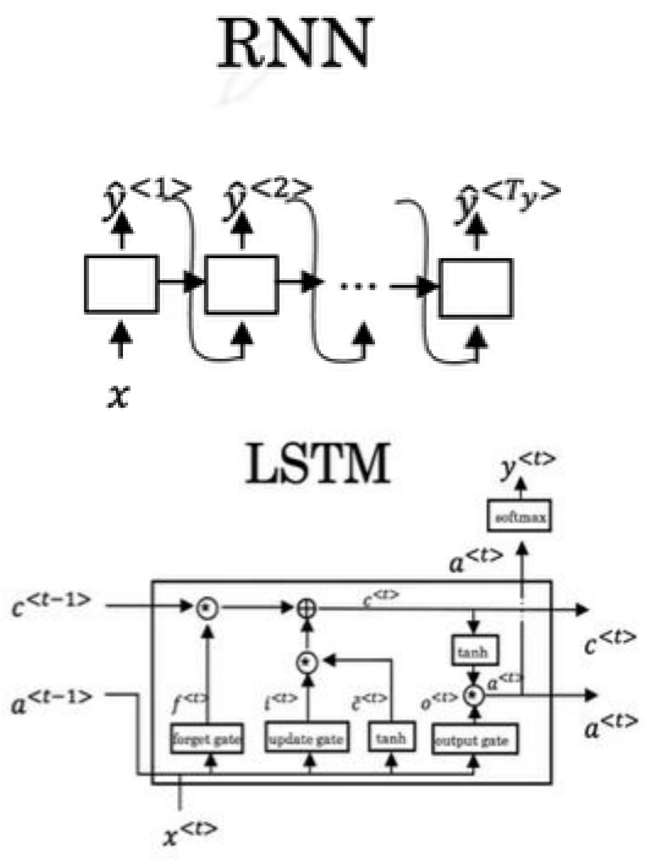
The decoder block appears again 6 times. The final output is transformed through a final linear layer and the output probabilities are calculated with the standard softmax function.

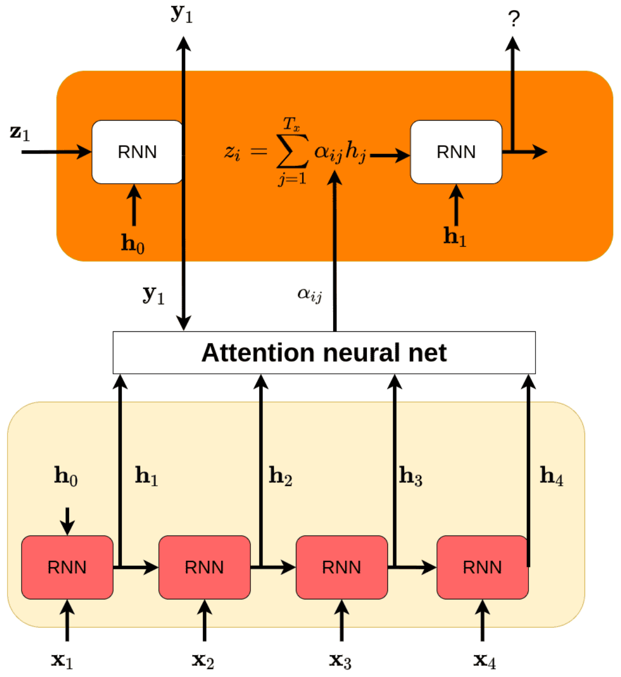
## Transformer [1]

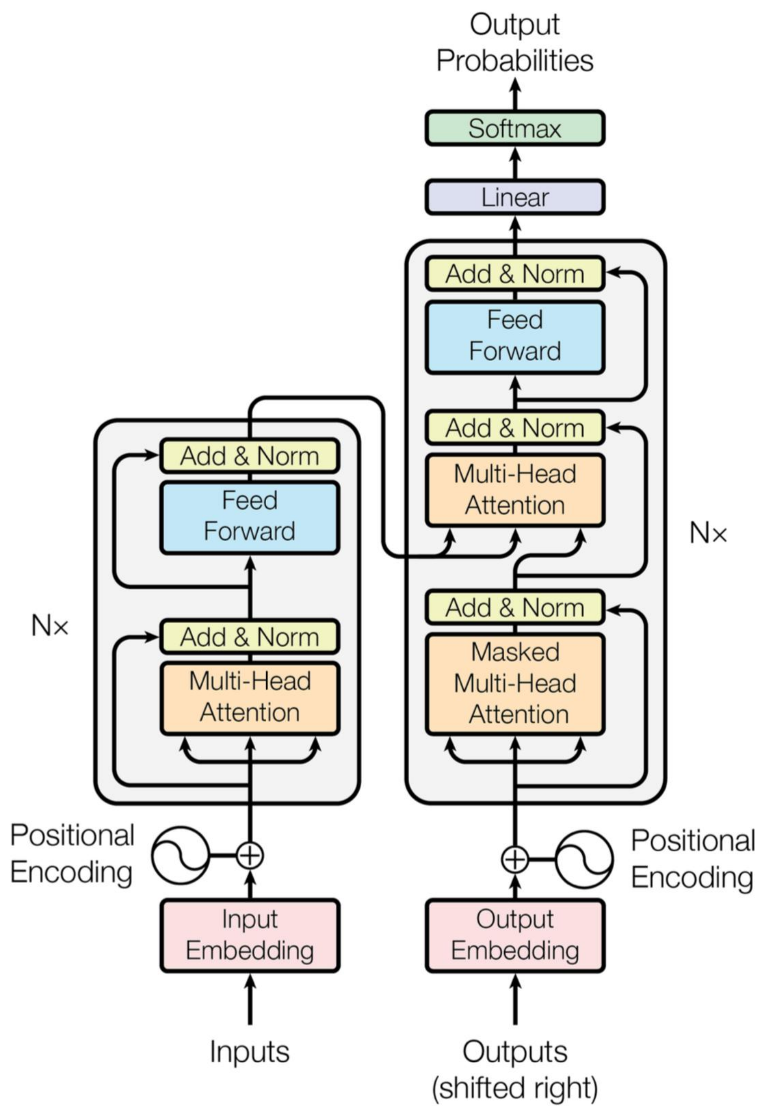


[1] Vaswani, Ashish et al. “Attention is All you Need.” (2017)

## Sequence models







## BLEU (bilingual evaluation understudy) [1]

- **BLEU** is an algorithm for evaluating the **quality of text** which has been machine-translated from one natural language to another.
- It's a metric, that measures the **precision** of **n-grams** (such as bigrams or trigrams) between the **predicted translation** and the **reference translations**

The children are sitting with their shoes off .

A man wearing a long white robe and a white cap , stands over a group of seated people .

A man in a white robe stands and talks to people who are sitting .

A man in a long white robe and white head cap is pointing to a group of people sitting on a blanket .

A man dressed in all white is talking to the people seated on the dirt floor .

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## N-gram

- It is just a fancy way of describing "a set **of 'n' consecutive words in a sentence"**.
- For instance, in the sentence **"The ball is blue"**, we could have n-grams such as:
  - **1-gram (unigram):** "The", "ball", "is", "blue"
  - **2-gram (bigram):** "The ball", "ball is", "is blue"
  - **3-gram (trigram):** "The ball is", "ball is blue"
  - **4-gram:** "The ball is blue"

The children are sitting with their shoes off .

A man wearing a long white robe and a white cap , stands over a group of seated people .

A man in a white robe stands and talks to people who are sitting .

A man in a long white robe and white head cap is pointing to a group of people sitting on a blanket .

A man dressed in all white is talking to the people seated on the dirt floor .

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## Precision

- Let’s say, that we have:
  - **Target Sentence:** He eats an apple
  - **Predicted Sentence:** He ate an apple
- We would normally compute the Precision using the formula:

$$\textbf{Precision} = \frac{\textbf{Number of correct predicted words}}{\textbf{Number of total predicted words}}$$

- Here **eats** and **ate** count as different words
- Precision = 3 / 4

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## Precision

- The issue is that only using precision allow us to cheat:
  - **Target Sentence:** He eats an apple
  - **Predicted Sentence:** He he he
- Precision = 3 / 3

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## Clipped Precision

- **Target Sentence 1: He eats a sweet apple**
- **Target Sentence 2: He is eating a tasty apple**
- **Predicted Sentence: He He He eats tasty fruit**

1. We compare each word from the predicted sentence with all of the target sentences.
2. If the word matches any target sentence, it is considered to be correct.
3. We limit the count for each correct word to the maximum number of times that that word occurs in the Target Sentence.

$$\text{Clipped Precision} = \frac{\text{Clipped \# of correct predicted words}}{\text{\# of total predicted words}}$$

| Word | *Matching Sentence* | *Matched Predicted Count* | *Clipped Count* |
|---|---|---|---|
| He | *Both* | *3* | *1* |
| eats | *Target 1* | *1* | *1* |
| tasty | *Target 2* | *1* | *1* |
| fruit | *None* | *0* | *0* |
| **Total** | | ***5*** | ***3*** |

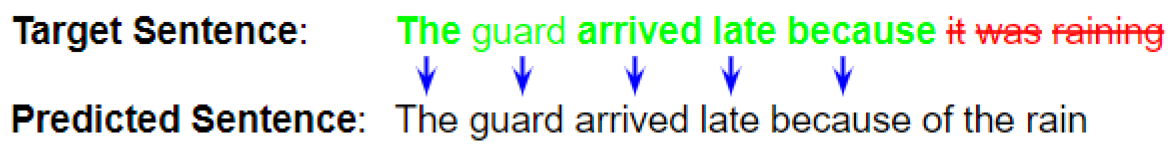
Clipped Precision = **3 / 6**

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## How is BLEU score calculated?

- **Target Sentence:** The guard arrived late because it was raining
- **Predicted Sentence:** The guard arrived late because of the rain

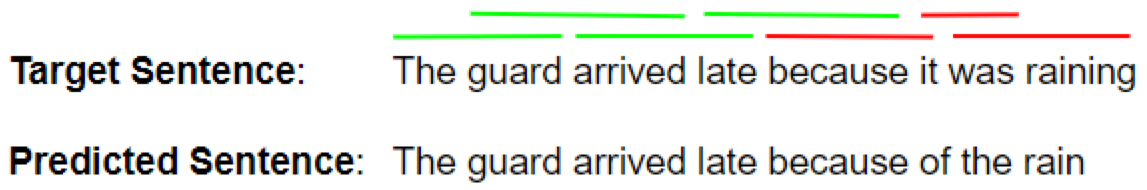
**Precision 1-gram:** $p_1 = \frac{5}{8}$



**Target Sentence:** The guard arrived late because ~~it was raining~~

**Predicted Sentence:** The guard arrived late because of the rain

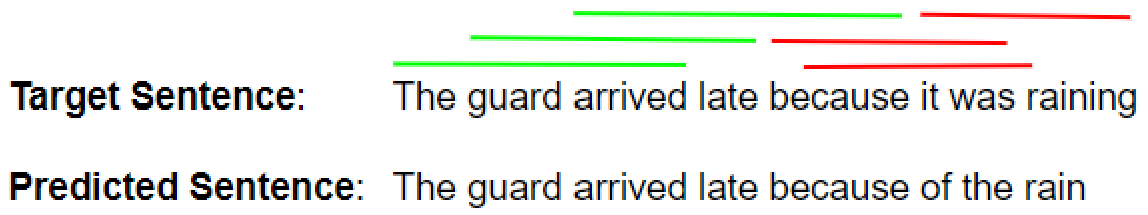
**Precision 2-gram:** $p_2 = \frac{4}{7}$



**Target Sentence:** The guard arrived late because it was raining

**Predicted Sentence:** The guard arrived late because of the rain

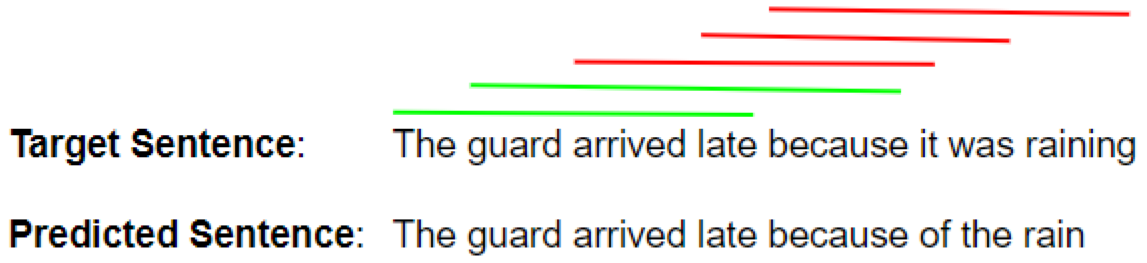
**Precision 3-gram:** $p_3 = \frac{3}{6}$



**Target Sentence:** The guard arrived late because it was raining

**Predicted Sentence:** The guard arrived late because of the rain

**Precision 4-gram:** $p_2 = \frac{2}{5}$



**Target Sentence:** The guard arrived late because it was raining

**Predicted Sentence:** The guard arrived late because of the rain

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## How is BLEU score calculated?

**Geometric Average Precision (GAP)**

- We combine these precision scores using the following formula

$$\mathbf{GAP}(N) = \prod_{n=1}^{N} p_n^w$$

- We usually use $N = 4$ and uniform weights $w = \frac{1}{N}$
- In case of $N = 4$ the Geometric Average Precision:

$$\mathbf{GAP}(N) = (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}}$$

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## How is BLEU score calculated?

**Brevity Penalty**

- The Brevity Penalty penalizes sentences that are too short.

$$\mathbf{BrevityPenalty} = \begin{cases} 1, & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)}, & \text{if } c \leq r \end{cases}$$

- $c$ is the predicted length -> number of words in the predicted sentence
- $r$ is the target length -> number of words in the target sentence (length closes to prediction length)

- In the previous example $c = 8$ and $r = 8$

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## How is BLEU score calculated?

- To get the BLEU score we multiply the **Brevity Penalty** with the **Geometric Average** of the **Precision Scores**

$$\mathbf{BLEU}(N) = \mathbf{BrevityPenalty} \cdot \mathbf{GAP}(N)$$

- **BLEU** Score can be computed for different values of N.
  - **BLEU-1** uses the **unigram** Precision score
  - **BLEU-2** uses the geometric average of **unigram** and **bigram** precision
  - **BLEU-3** uses the geometric average of **unigram**, **bigram**, and **trigram** precision
  - and so on.

[1] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b/

## Natural Language Processing Tasks

### Name Entity Recognition

- A text corpus tagged with 4 different entity types
  - PER (persons), LOC (locations), ORG (organization), MISC (miscellaneous)

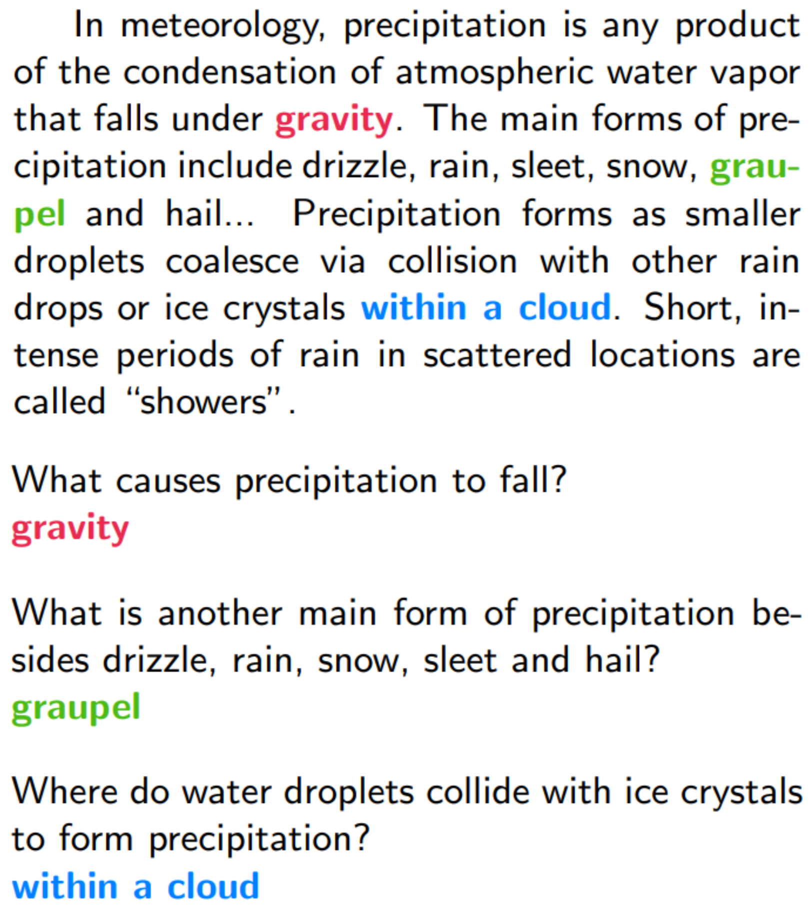
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Input X: | Harry | Potter | and | Hermione | Granger | invented | a | new | spell |
| Tags: | B-PER | I-PER | O | B-PER | I-PER | O | O | O | O |
| Output Y: | 1 | 2 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Input X: | The | European | Commission | said | on | Thursday | it | disagreed | with | German | advice |
| Tags: | O | B-ORG | I-ORG | O | O | O | O | O | O | B-MISC | O |
| Output Y: | 0 | 3 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 |

## Natural Language Processing Tasks

### Question Answering

- Stanford Question Answering Dataset (SQuAD)
- **Context**
- **Question – Answer** pairs
- The answer is a span in a given Wikipedia paragraph



| | |
|---|---|
| Context | In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers". |
| Question 1 | What causes precipitation to fall? |
| Answer 1 | **gravity** |
| Question 2 | What is another main form of precipitation besides drizzle, rain, snow, sleet and hail? |
| Answer 2 | **graupel** |
| Question 3 | Where do water droplets collide with ice crystals to form precipitation? |
| Answer 3 | **within a cloud** |

## Natural Language Processing Tasks

### Textual Entailment

Stanford Natural Language Inference (SNLI)

Determining whether a "**hypothesis**" is true, given a "**premise**".

Five different annotator – **Gold label**

**Contradiction** - the hypothesis cannot be true given the premise.

**Neutral** - the hypothesis might be true or false; it's not entailed or contradicted.

**Entailment** - the hypothesis must be true given the premise.

| Text | Judgements | Hypothesis |
|---|---|---|
| A man inspects the uniform of a figure in some East Asian country. | contradiction C C C C C | The man is sleeping |
| An older and younger man smiling. | neutral N N E N N | Two men are smiling and laughing at the cats playing on the floor. |
| A black race car starts up in front of a crowd of people. | contradiction C C C C C | A man is driving down a lonely road. |
| A soccer game with multiple males playing. | entailment E E E E E | Some men are playing a sport. |
| A smiling costumed woman is holding an umbrella. | neutral N N E C N | A happy woman in a fairy costume holds an umbrella. |

## Natural Language Processing Tasks

### Machine Translation

- The name speaks for itself
- BUT, **input – output** pairs in **source – target** language.
- For one input there might be **multiple candidate** translations

| English | French |
|---|---|
| It's cold. | C'est froid. |
| Stay calm. | Reste tranquille. |
| Stop that. | Arrêtez ça ! |
| I will submit the assignment tonight | Je rendrai le devoir ce soir. |

# Generative Pre-trained Transformer (GPT) – June 2018

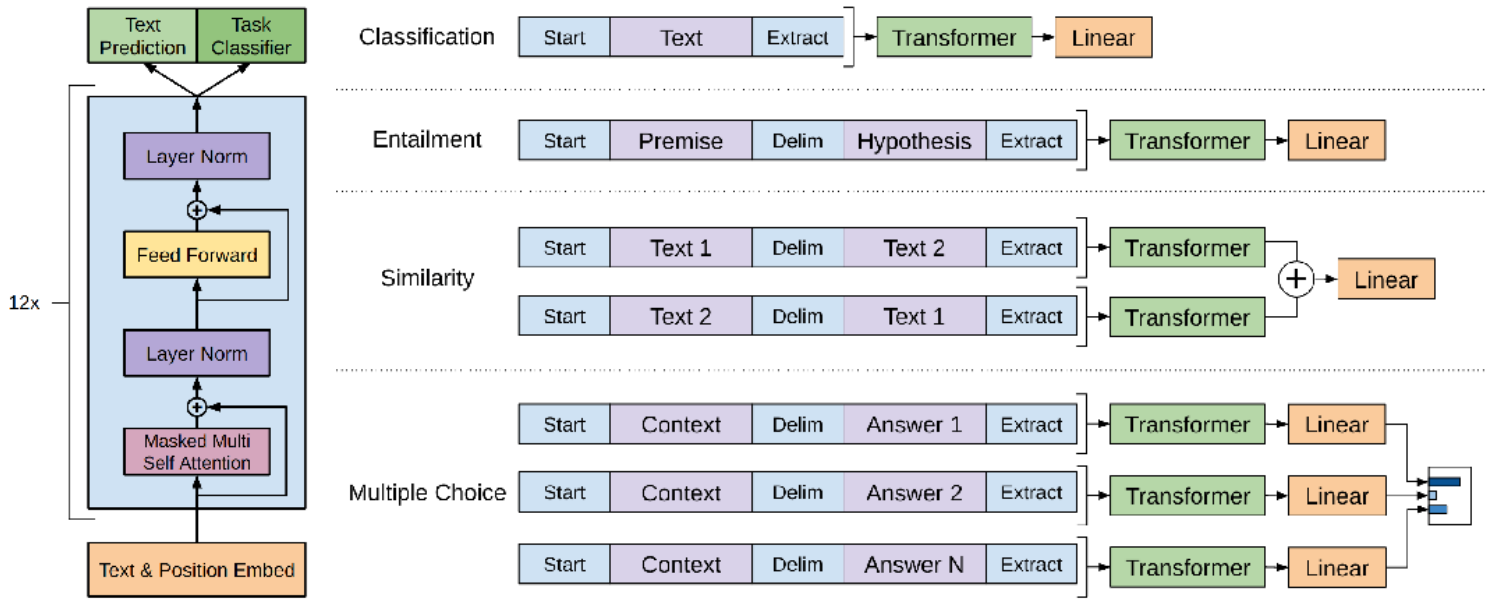
The original goal is Language Modelling (LM)

Uses Masked Self-Attention to limit the attention to the previous tokens only (left-to-right)

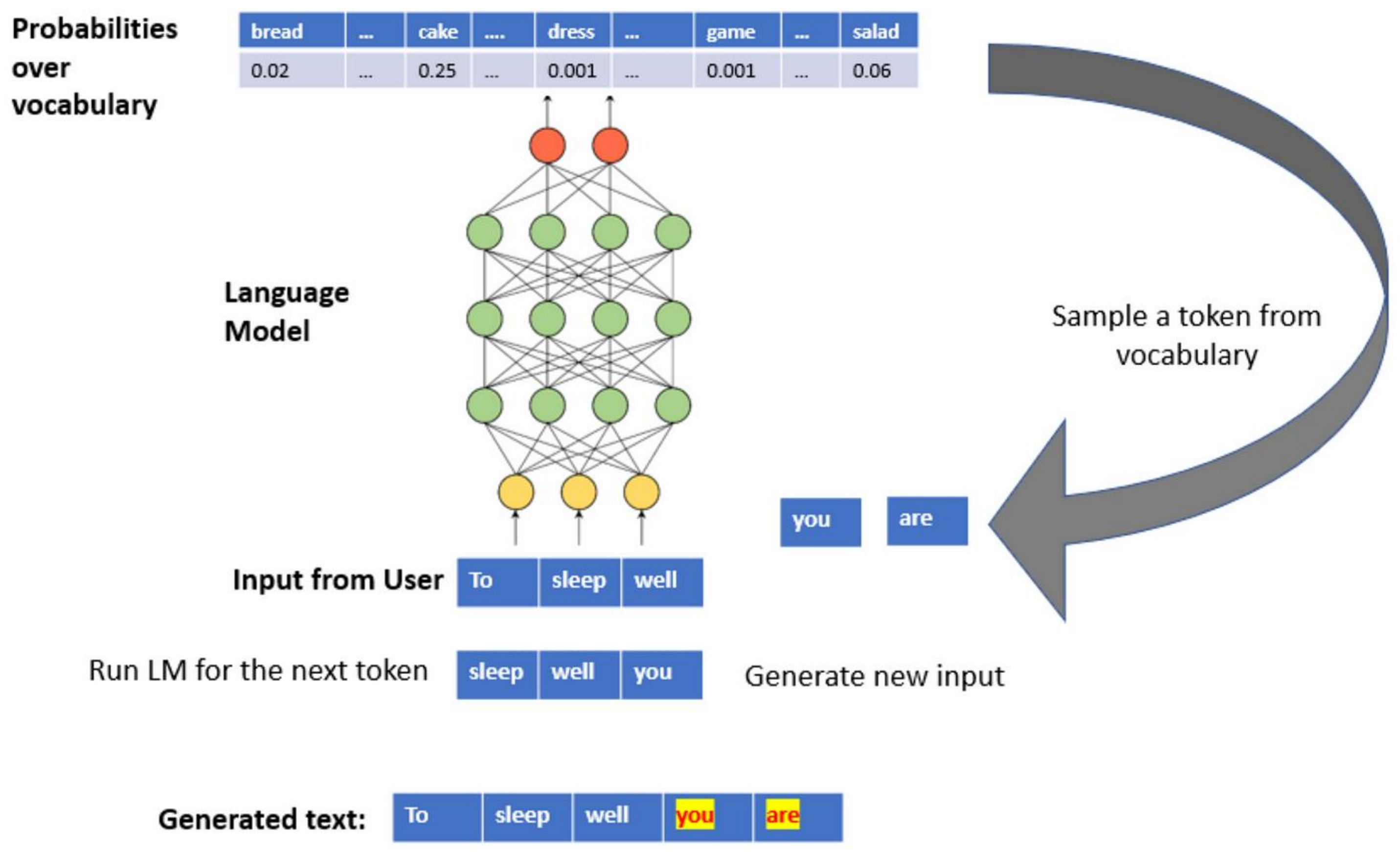
Two stage training:

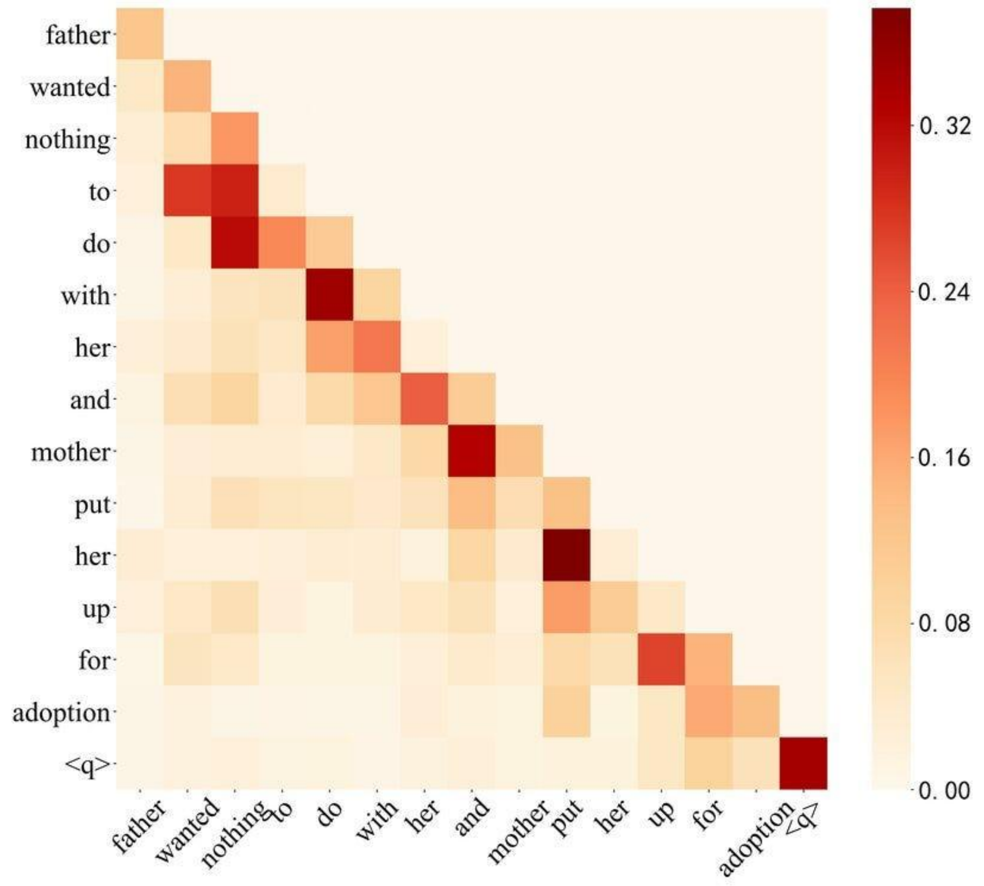
1. Unsupervised pre-training:
   - The goal is to predict the next token based on the previous tokens.

2. Supervised fine-tuning:
   - Predict the label (y) based on the input tokens



## Generative Pre-trained Transformer (GPT) – June 2018

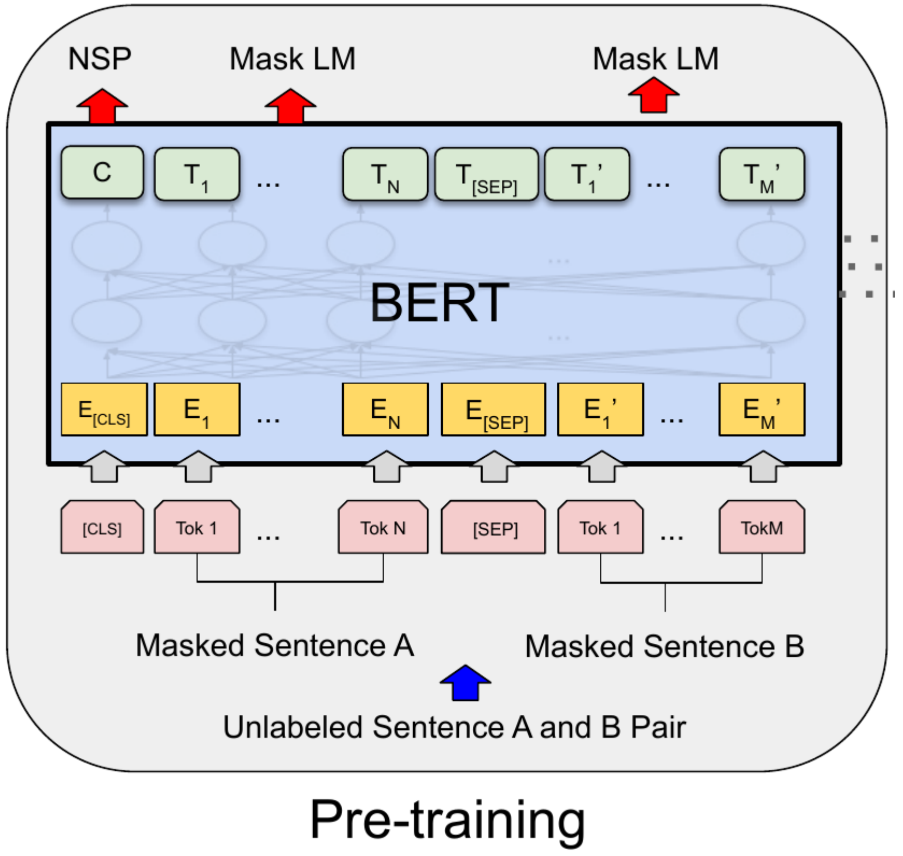


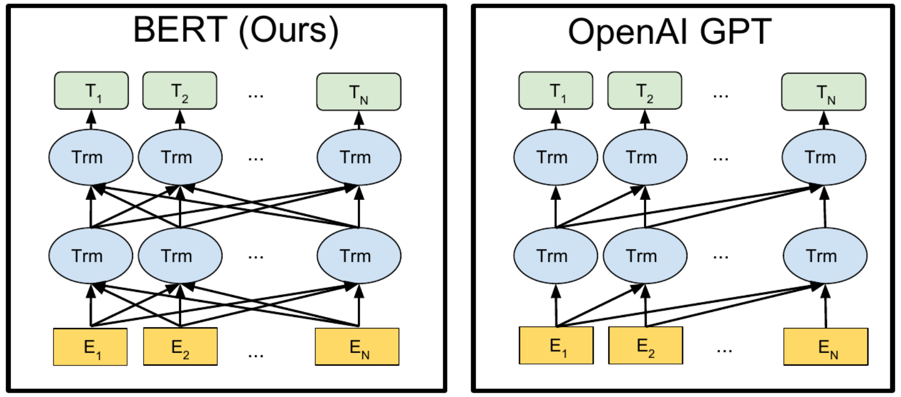


## Bidirectional Encoder Representations from Transformers (BERT) – May 2019

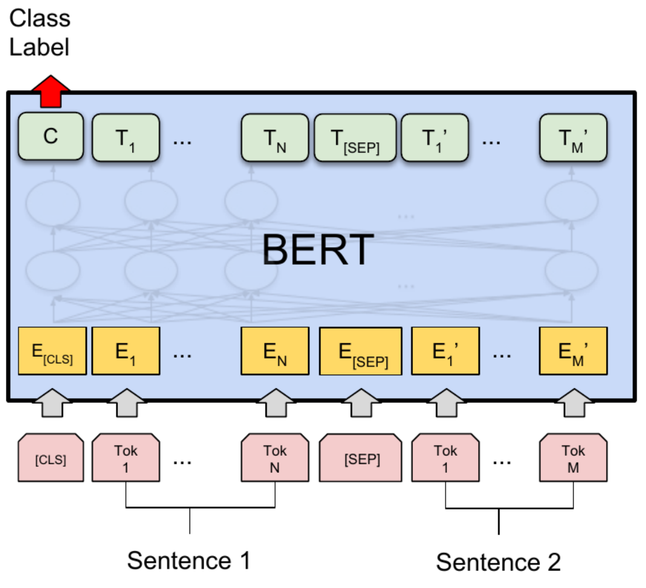
- Compared to OpenAI GPT it uses a bidirectional self-attention
- Trained on 2 tasks at the same time during pre-training
  1. Masked LM (15% of the tokens are masked)
  2. Next Sentence Prediction

- A special [CLS] token is introduced at the beginning of each sequence.

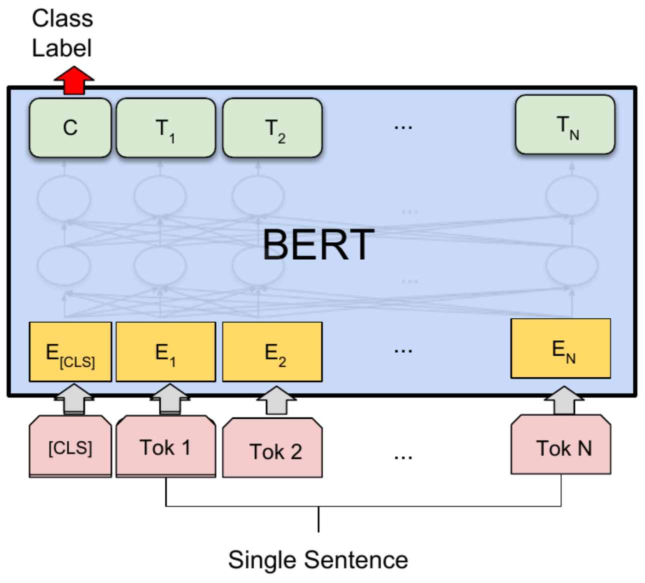




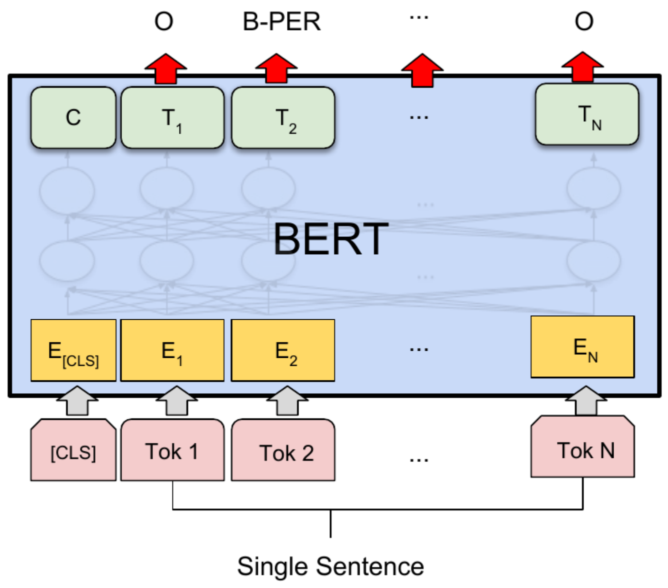
## Bidirectional Encoder Representations from Transformers (BERT) – May 2019



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

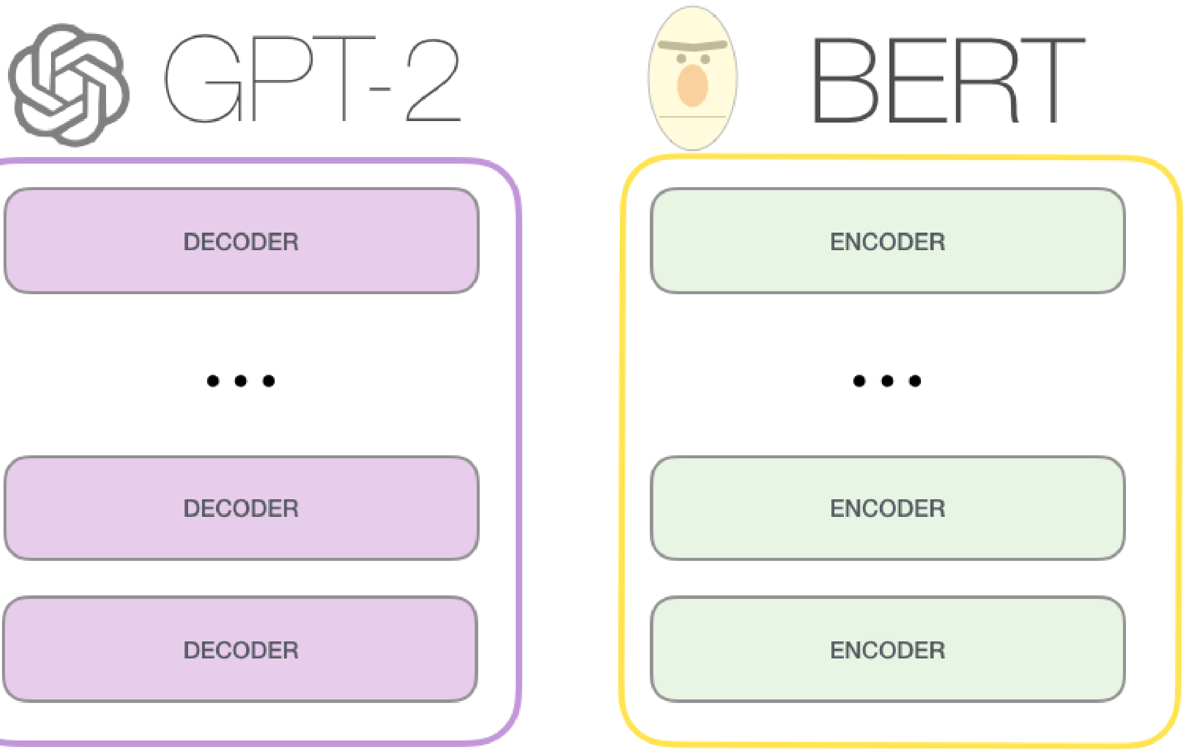


(b) Single Sentence Classification Tasks: SST-2, CoLA



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

## Transformer Architectures



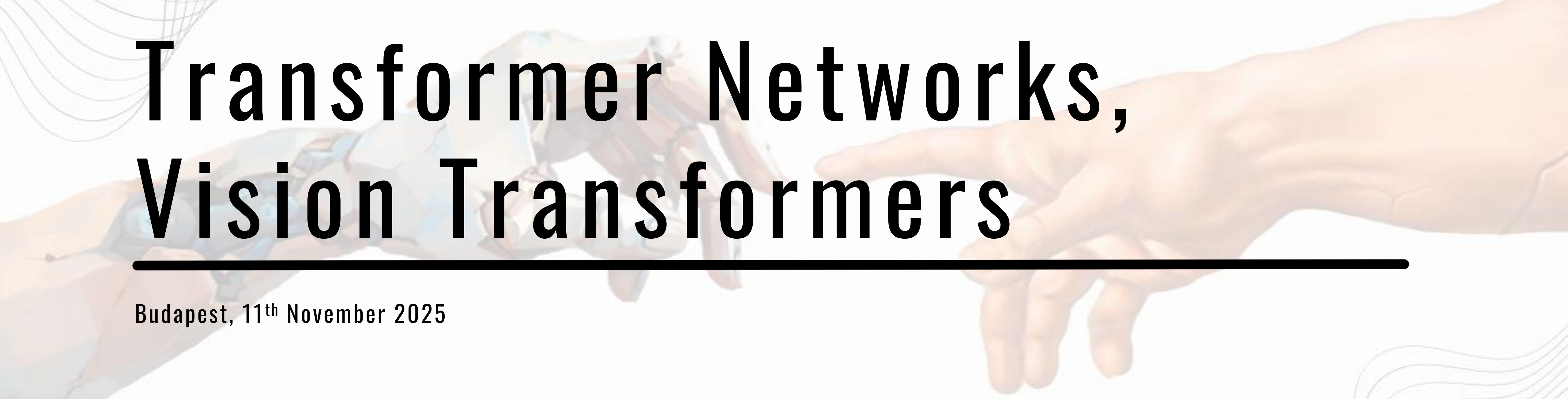
Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

# Lecture 10.

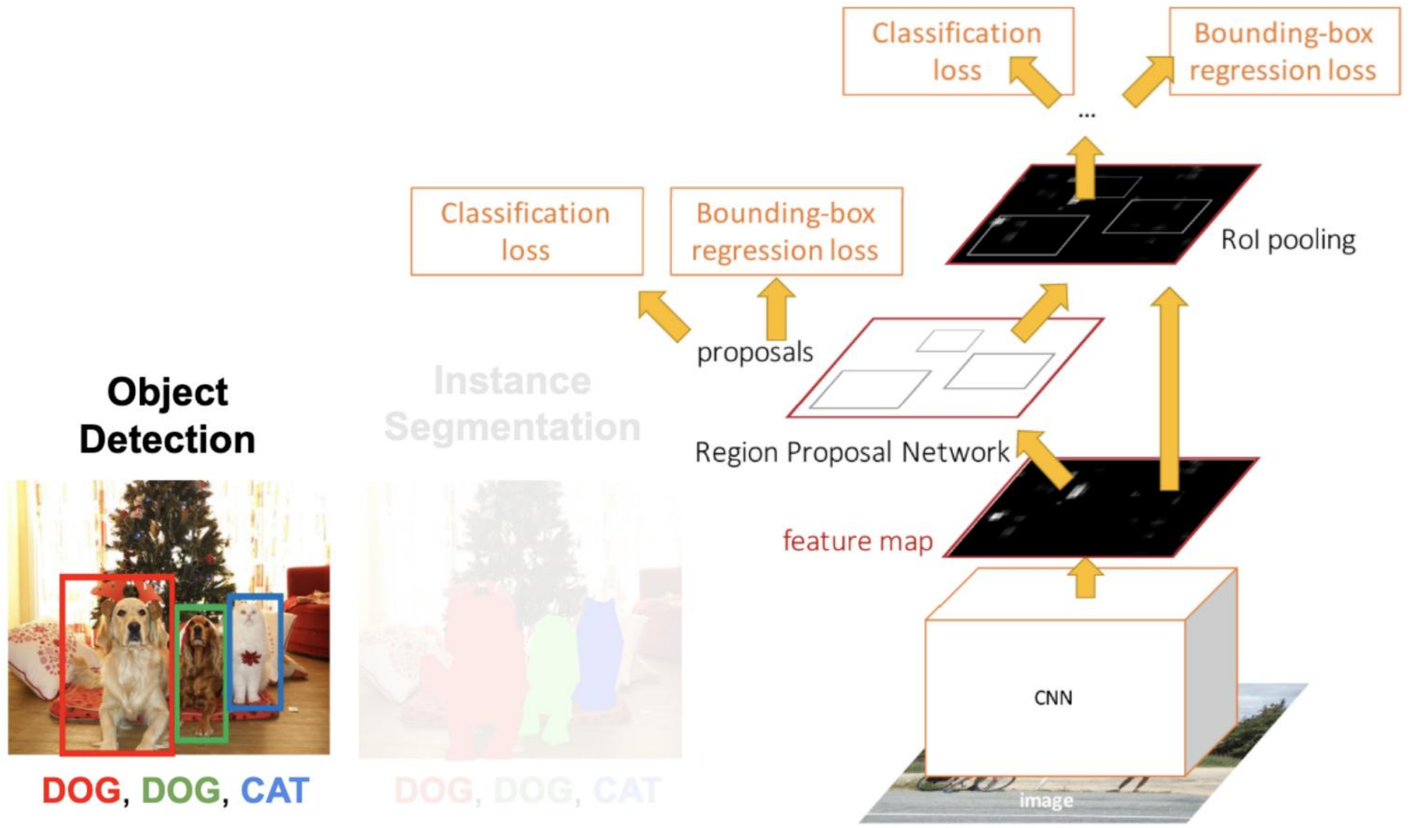
# Transformer Networks, Vision Transformers

Budapest, 11th November 2025



## Object detection

### Object Detection: Faster R-CNN [4]



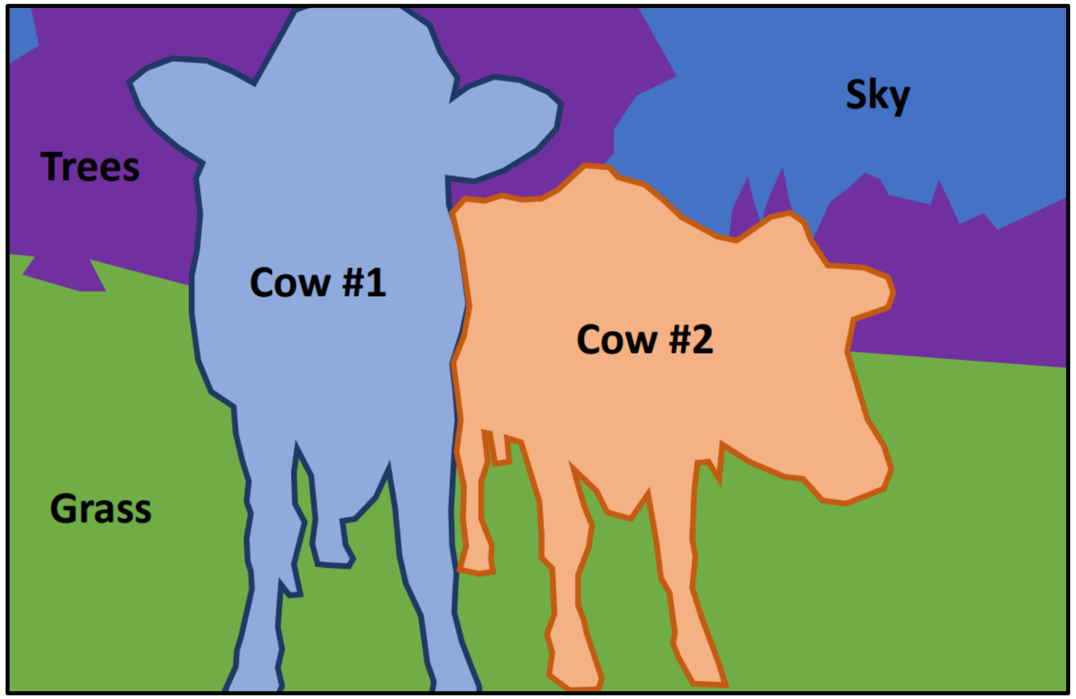
[4] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

## Instance Segmentation

Beyond Instance Segmentation: Panoptic Segmentation [3]

Label all pixels in the image (both things and stuff).

For "thing" categories also separate into instances.

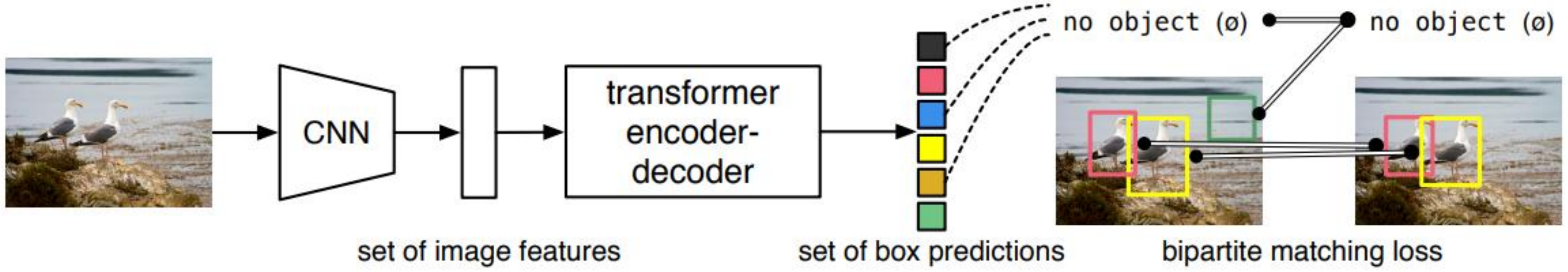


[3] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.00868

## DEtection TRansformer (DETR) – May 2020 [1]

End-to-End Object Detection with Transformers - arxiv.org/abs/2005.12872

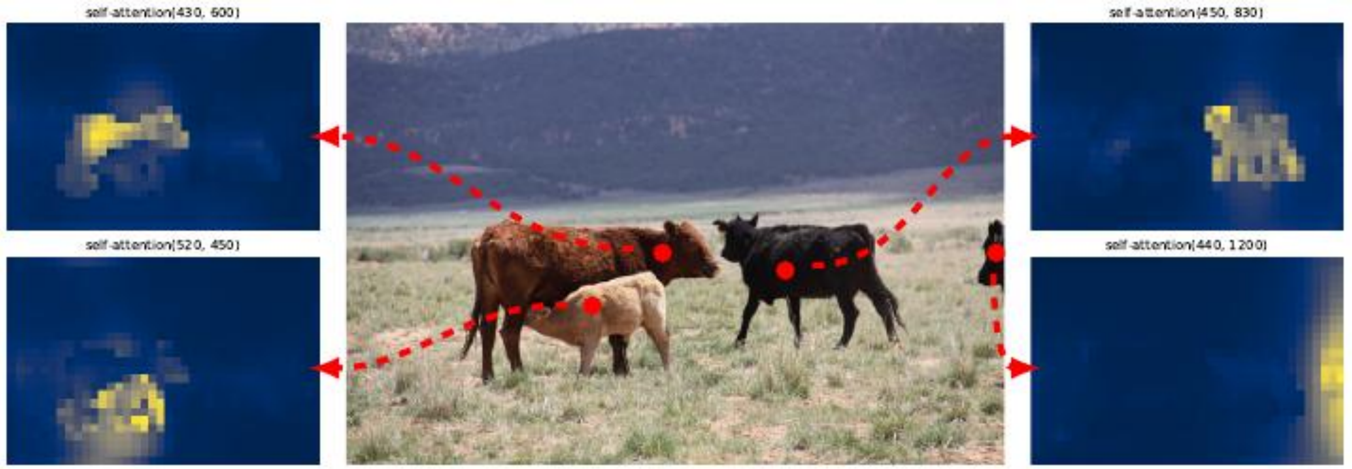
Simple architecture:
1. CNN backbone
2. Encoder-Decoder Transformer
3. Feed Forward Network

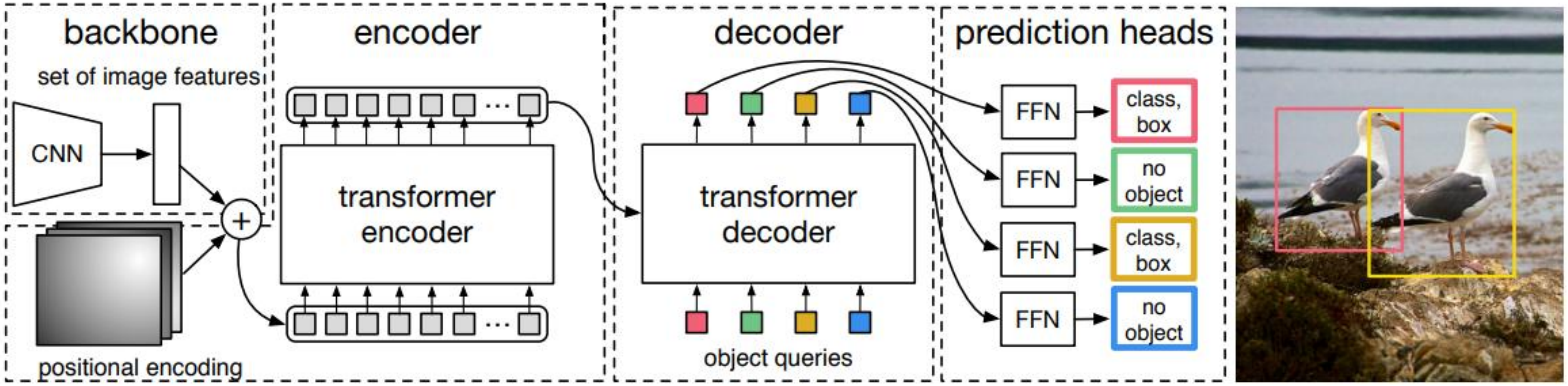


[1] End-to-End Object Detection with Transformers - arxiv.org/abs/2005.12872

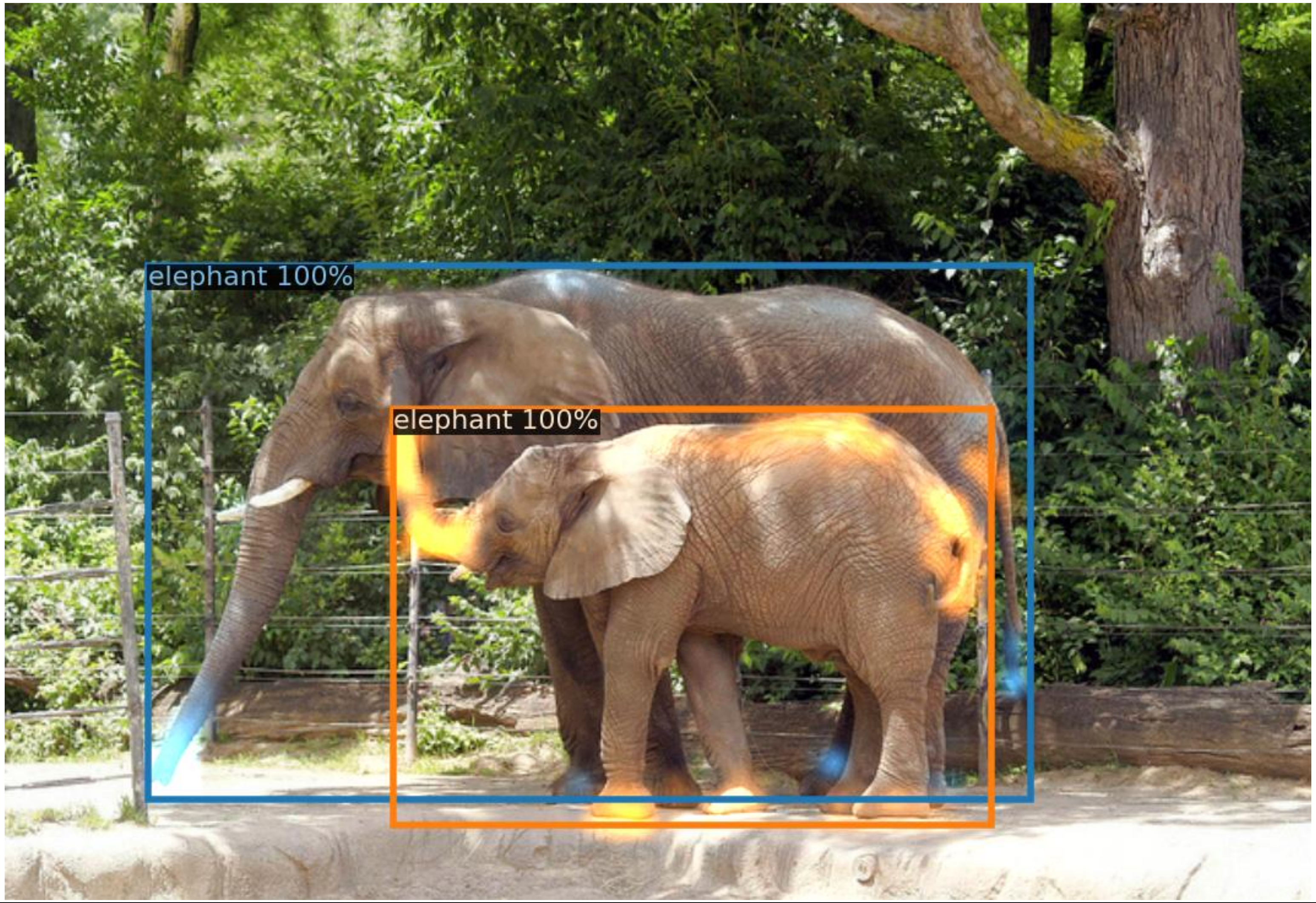
## DEtection TRansformer (DETR) – May 2020

1. Does N object detection parallelly based on learned object queries – (contrary to Region Proposals)
2. Uses a bipartite matching loss to remove duplicate bounding boxes – (contrary to Non-Maximum Suppression)



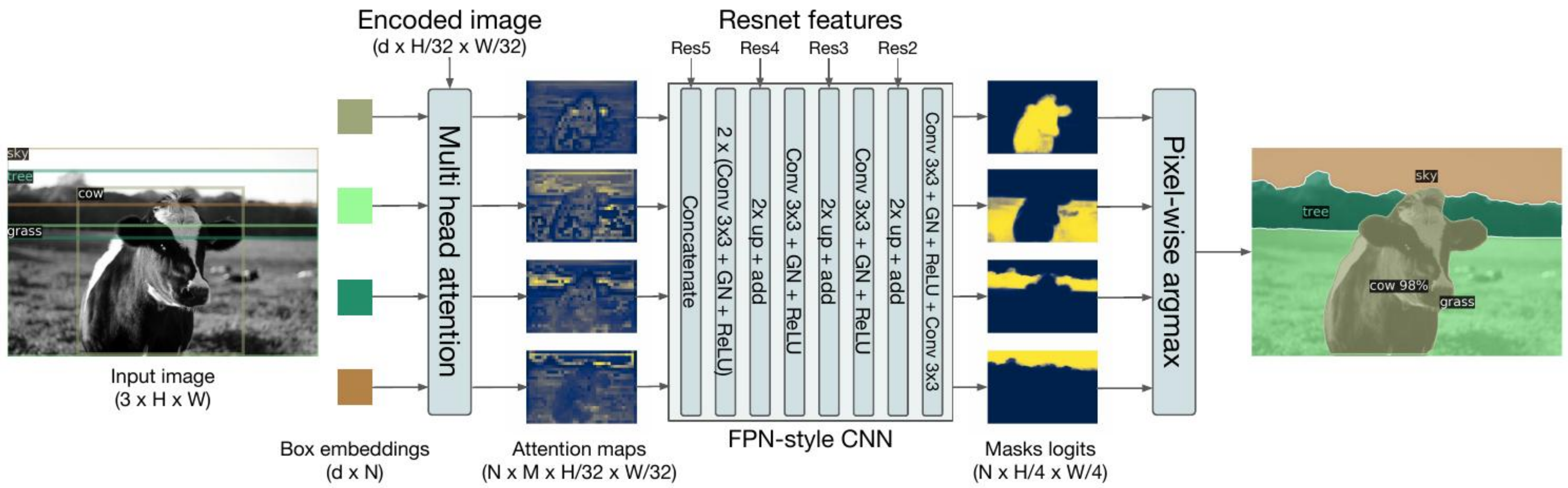


## DEtection TRansformer (DETR) – May 2020



## DEtection TRansformer (DETR) – May 2020

- DETR can be naturally extended by adding a mask head on top of the decoder outputs.
- Then it will generate the masks parallel to each predicted object



## Vision Transformers – June 2021 [2]

- *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* - *arxiv.org/abs/2010.11929*

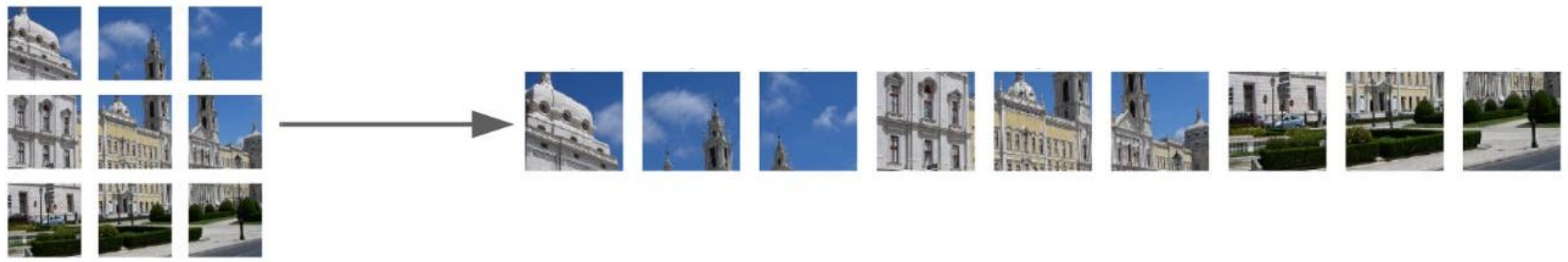
- In vision, attention is either applied in conjunction with convolutional networks or used to replace certain components of convolutional networks while keeping their overall structure in place.
- Reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks

- Transformers operate on a sequence of tokens
- How do we transform an image into tokens?

[2] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale - arxiv.org/abs/2010.11929

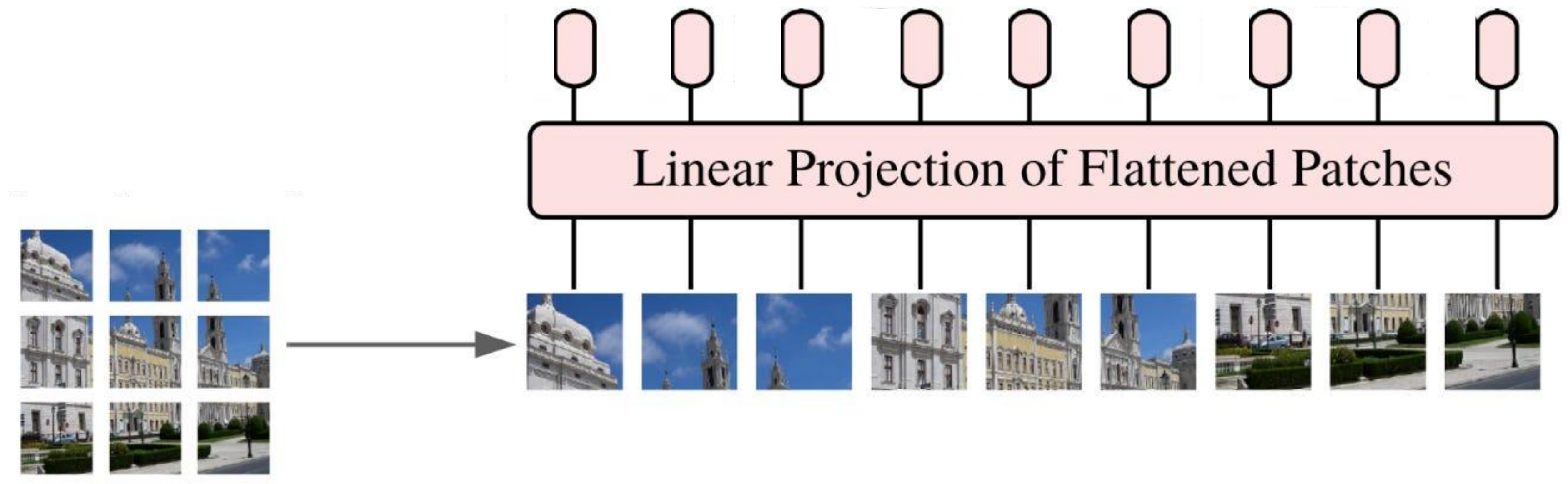
## Image Tokenization

1. Reshape images of $x \in \mathbb{R}^{H \times W \times C}$ into $N = \frac{HW}{P^2}$ patches
   - (H, W) – image resolution
   - C – number of channels
   - (P, P) – patch resolution



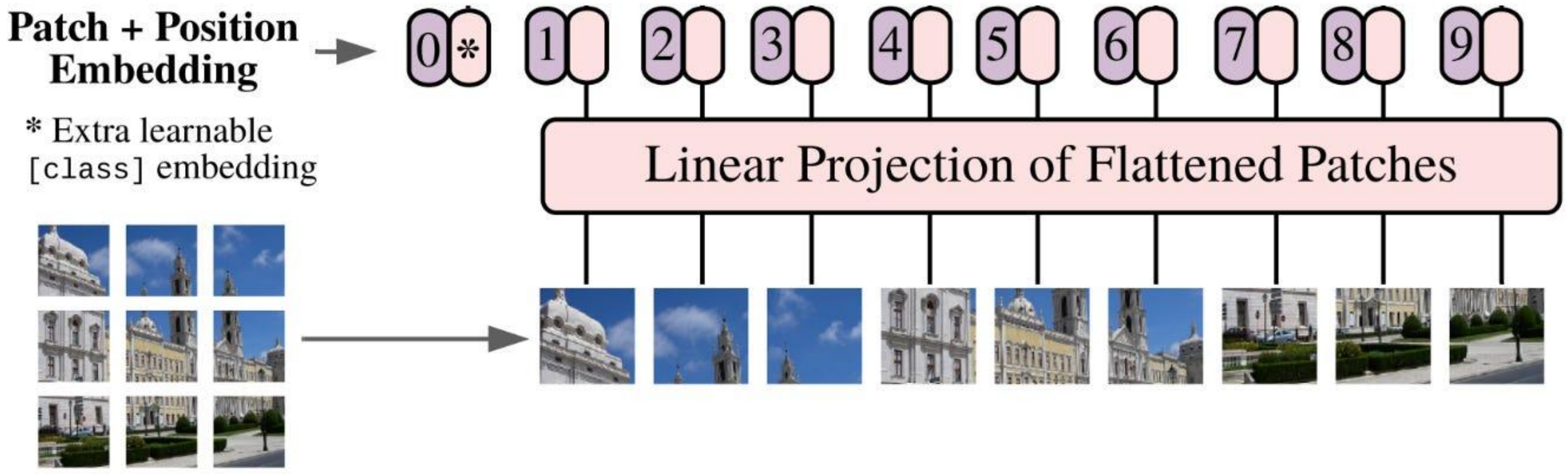
## Image Tokenization

2. The Transformer uses constant latent vector size $D$ through all of its layers
    - Flatten all the patches and apply a learnable linear projection (*patch embeddings*)



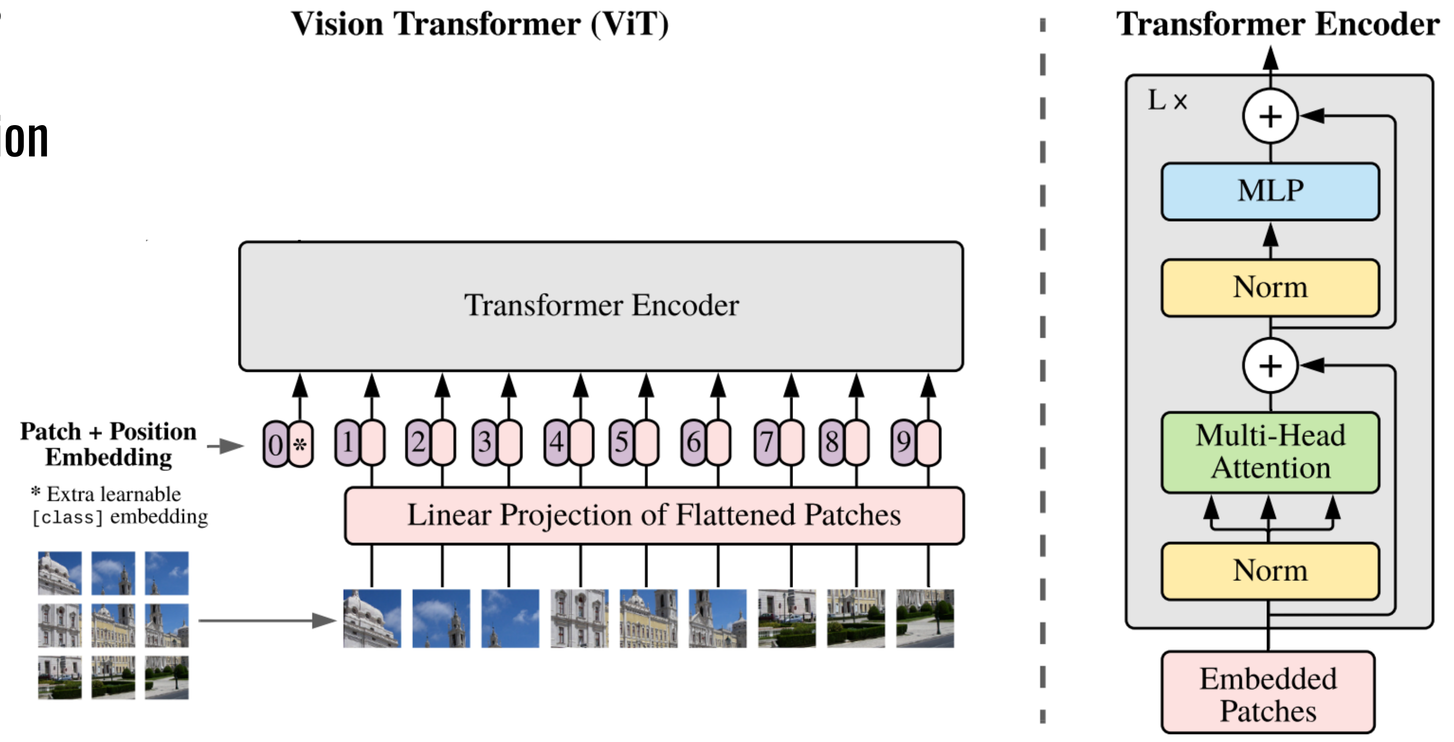
## Token Processing

3. Processing the tokens
    - Prepend a learnable embedding to the patch embeddings
    - Apply position embedding


Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

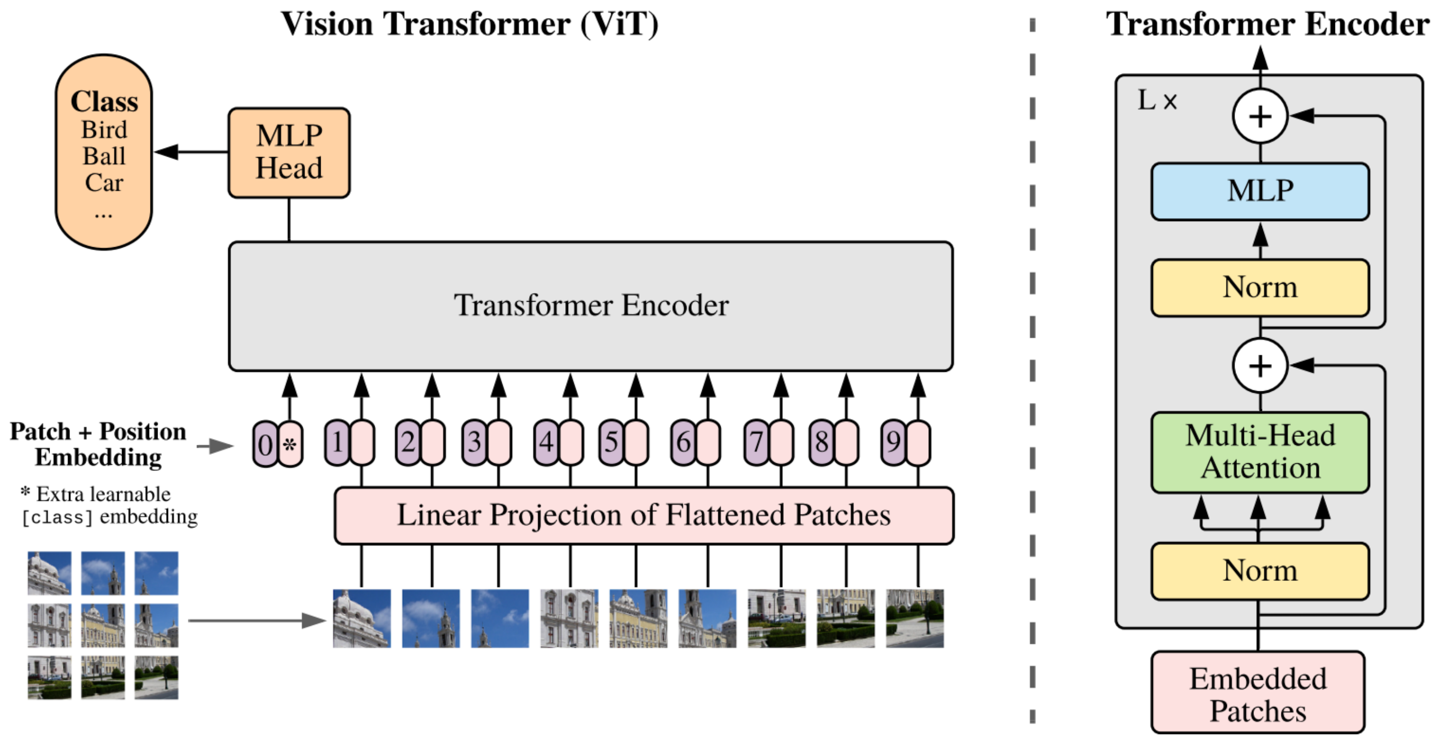
## Encoder Block

4. Feeding the embedded patches to the Transformer Encoder
   - Input the sequence of embedded patches $([\boldsymbol{z_0^0}, z_0^1, \dots, z_0^N])$
   - At the end we get the image representation $([\boldsymbol{z_L^0}, z_L^1, \dots, z_L^N])$



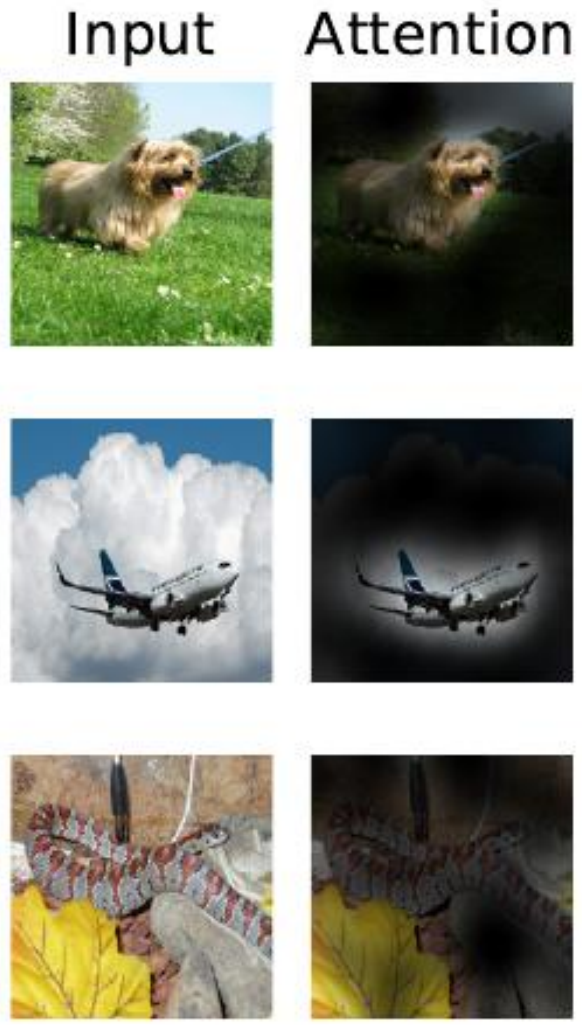
# Prediction Processing

5. Classification MLP head
    - Attaching a classifier head to $z_L^0$



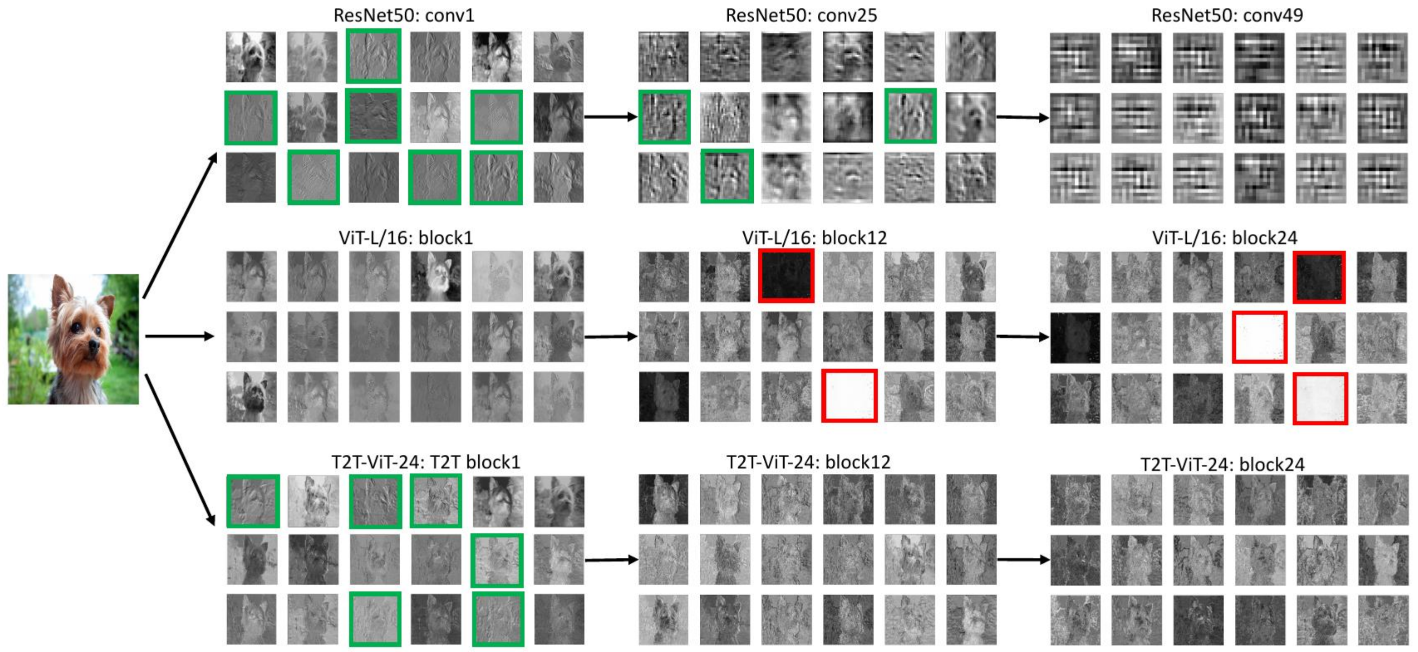
## Comparison with Convolutional Networks

- ViT has much less image-specific inductive bias
  - features like **edges**, **textures**, and **patterns** are spatially localized and translationally invariant
- In CNNs, locality, two-dimensional neighbourhood structure, and translation equivariance are baked into the whole model
- Position embedding does not carry information about the 2D position of the patches



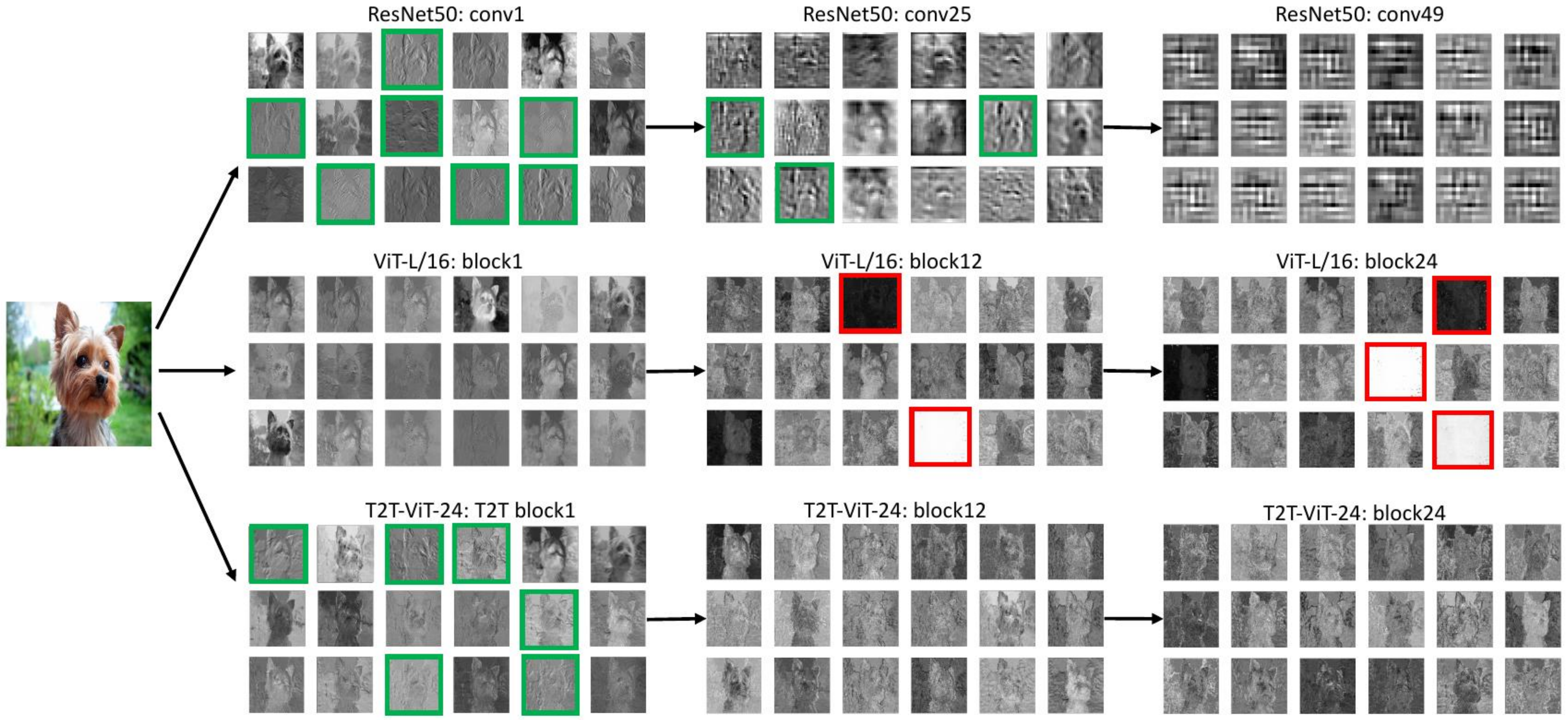
## Token-to-Token ViT (T2T-ViT) – Nov 2021

- ViT achieves inferior performance to CNNs when trained on a midsize dataset
  1. The tokenization fails to model the important local structure such as edges, lines, etc.
  2. The redundant attention backbone leads to limited feature richness



[3] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale - arxiv.org/abs/2010.11929

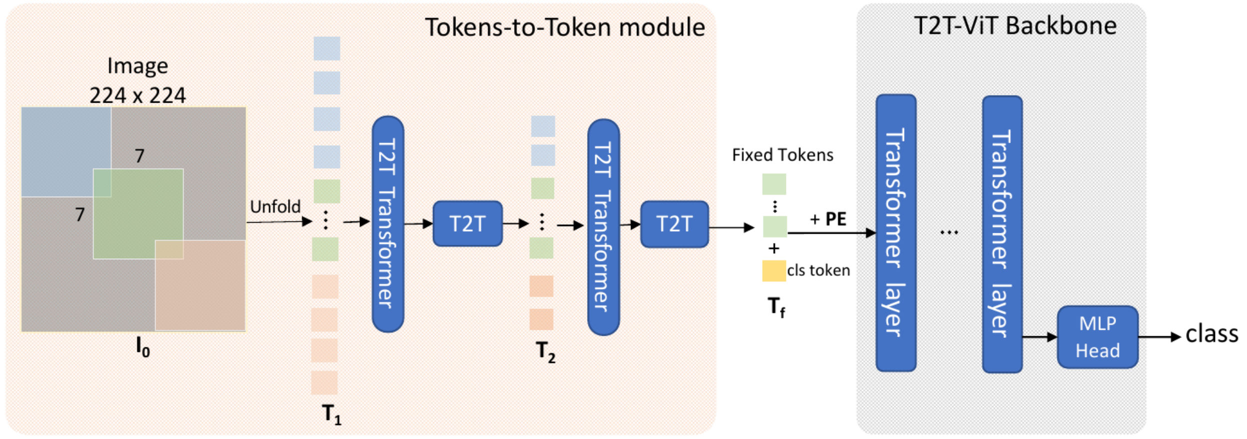
## Token-to-Token ViT (T2T-ViT)

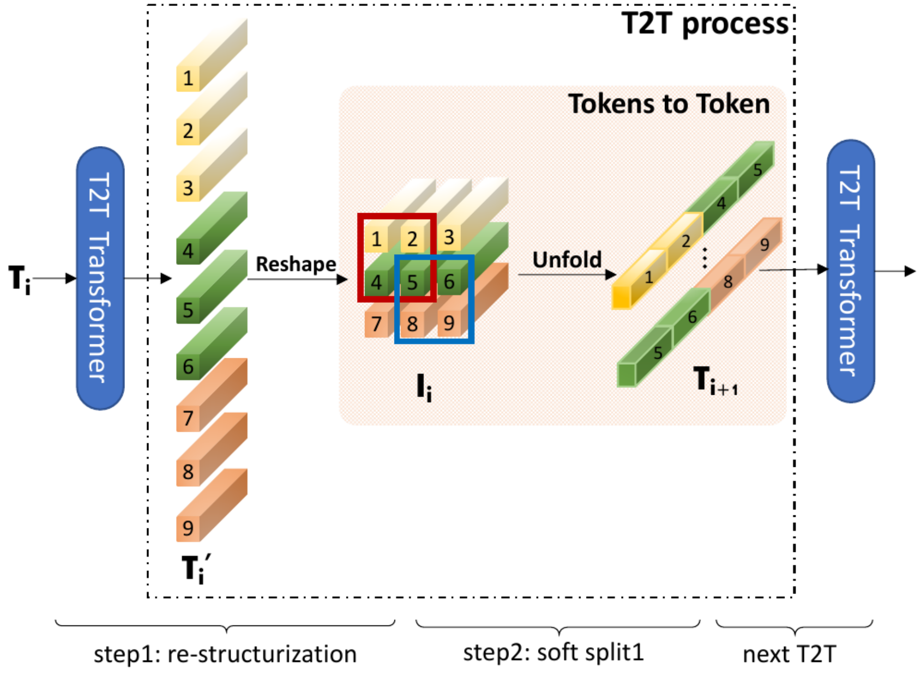


# Token-to-Token ViT (T2T-ViT)

1. A layer-wise "Token-to-token module"
2. An efficient "T2T-ViT backbone"

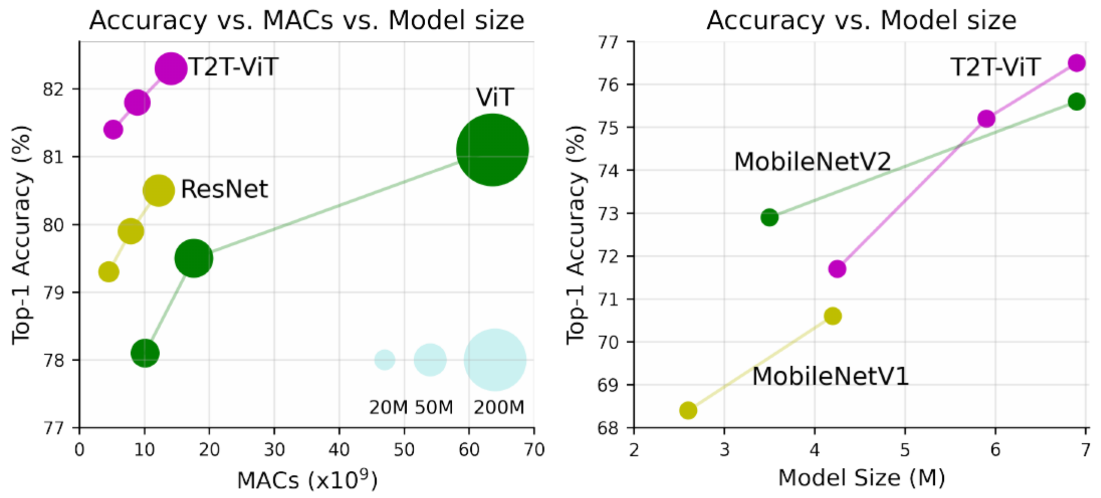
- The generated tokens are reordered like an "image"
- Then areas closer together are grouped together into a new token





## Token-to-Token ViT (T2T-ViT)

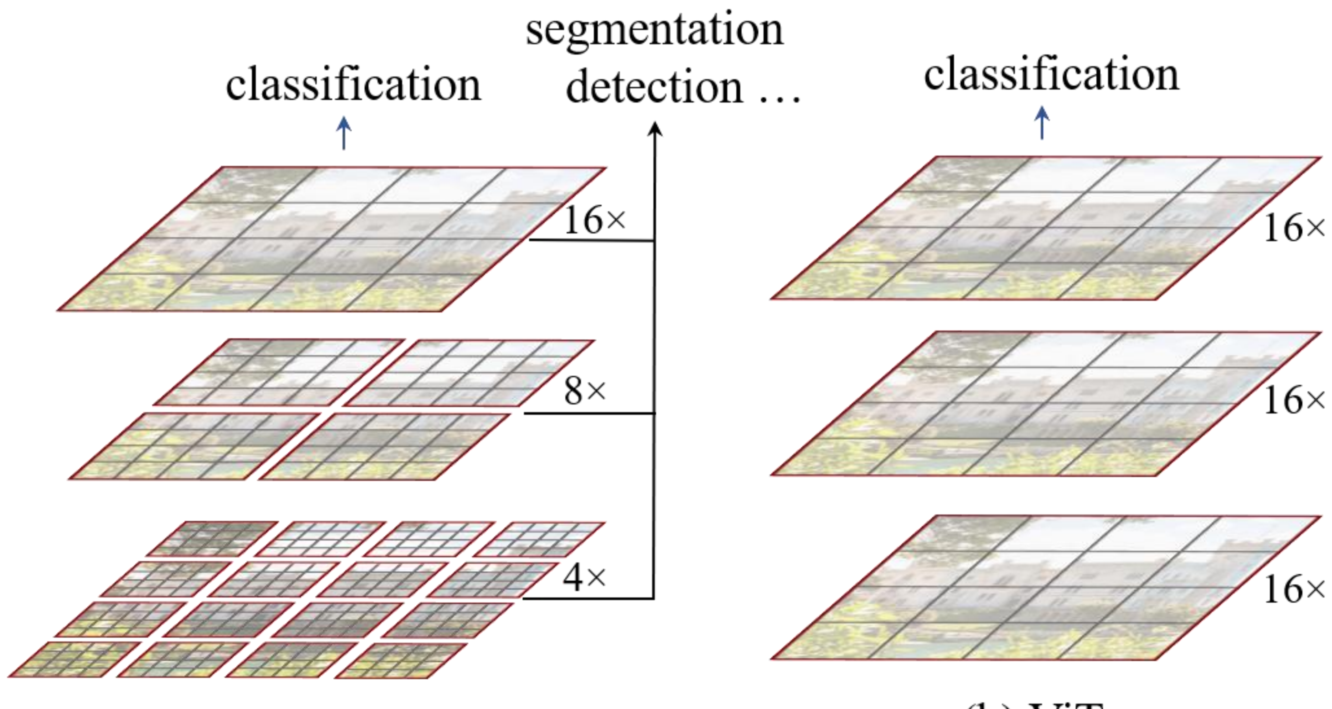
- While "vanilla" ViT requires a large dataset and more tuneable parameters to beat the "state-of-the-art" (**JFT-300M**) CNN models
- T2T-ViT requires smaller datasets and less tuneable parameter



## Swin Transformer – Aug 2021 [4]

- ViT - computational complexity is quadratic to image size.
  - Conduct global self attention, where the relationships between a token and all other tokens are computed

- Calculate self-attention within local windows
- Patch merging and scaling as the network gets deeper
- Shifted window approach with fixed number of patches.

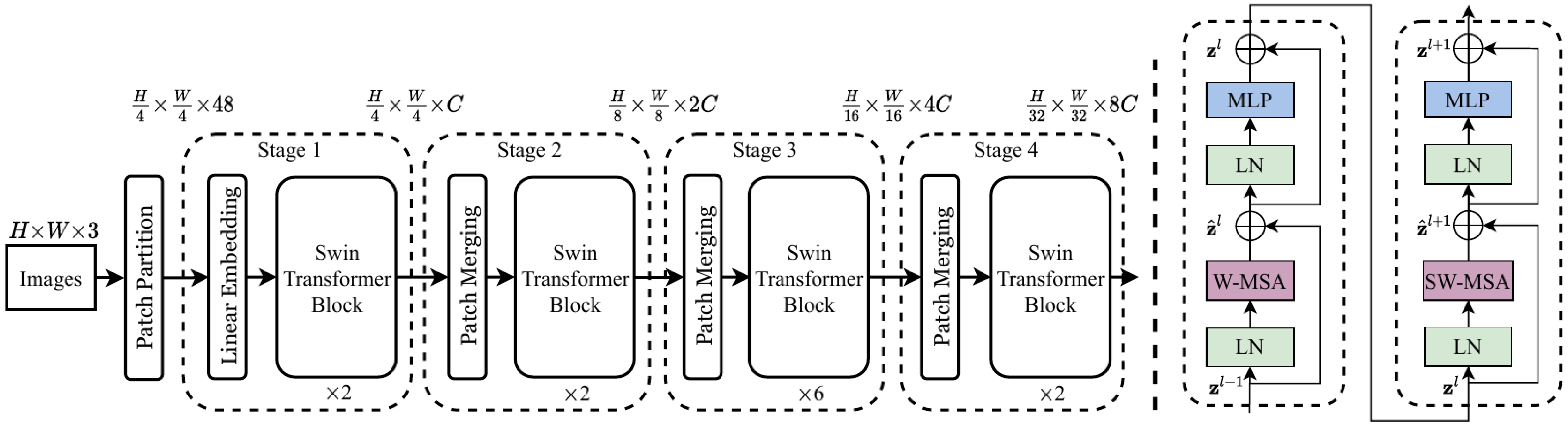


(a) Swin Transformer (ours) (b) ViT

[4] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows - https://arxiv.org/abs/2103.14030

## Swin Transformer – Aug 2021 [4]

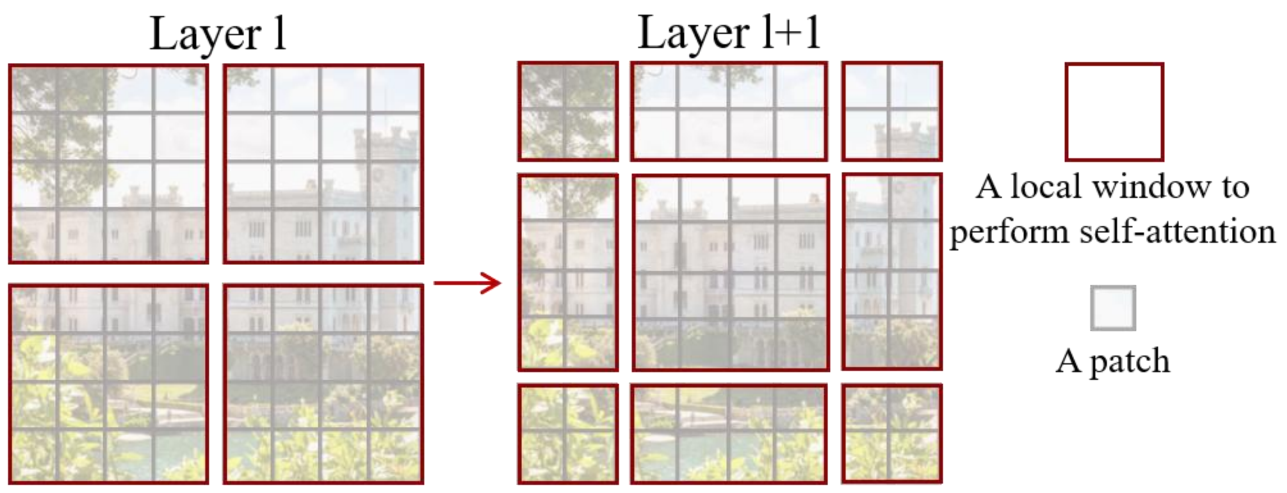
- The number of tokens reduced in the patch merging layer
  - Produce hierarchical representations
  - Similar to Feature Pyramid Networks

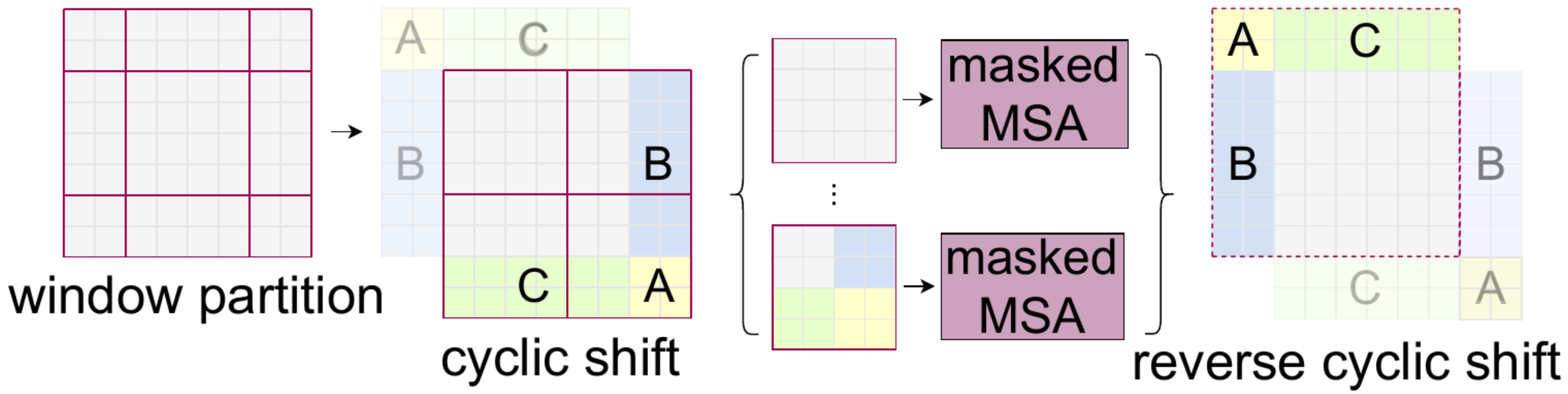


(a) Architecture (b) Two Successive Swin Transformer Blocks

## Swin Transformer – Aug 2021 [4]

- Shifted window partitioning to introduce cross-window connection between non-overlapping windows.
- Alternates between two configuration.
- Masking mechanism - limit self-attention non-adjacent windows





## Further Links + Resources

- https://theaisummer.com/attention/
- https://theaisummer.com/transformer/
- https://jalammar.github.io/illustrated-transformer/
- https://nlp.seas.harvard.edu/annotated-transformer/ !!!!
- https://www.youtube.com/watch?v=bCz4OMemCcA
- Coursera Deep Learning Specialization

Please read this:

- Beam search: https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24
- BLEU score: https://cloud.google.com/translate/automl/docs/evaluate#bleu

## Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

# That’s all for today!