ieeta

Departamento de Eletrónica, Telecomunicações e Informática

Complements of Machine Learning

Lecture 9 : Attention Mechanism & Transformers

Petia Georgieva
(petia@ua.pt)

universidade de aveiro

Outline

1. Language model - greedy search, Beam Search

2. Neural Machine Translation with Attention - Attention Model Architecture

3. Speech Recognition –
CTC (Connectionist Temporal Classification) cost

4. Transformers

5. Input & Positional Embedding

6. Multi-head Attention

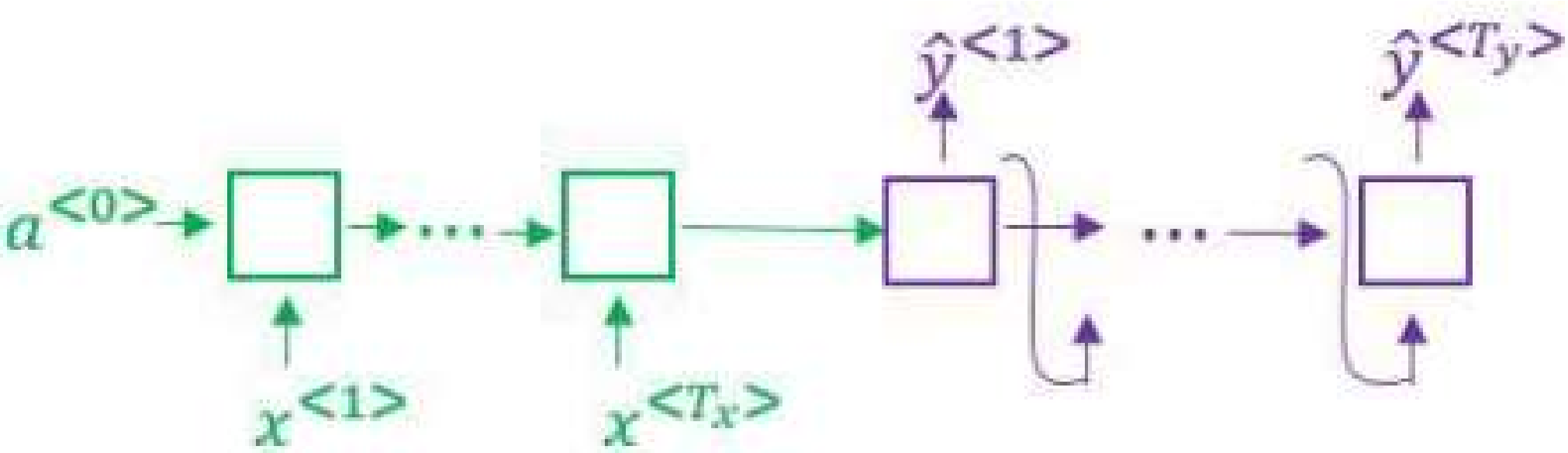
Sequence to Sequence architectures

Machine Translation or Speech Recognition

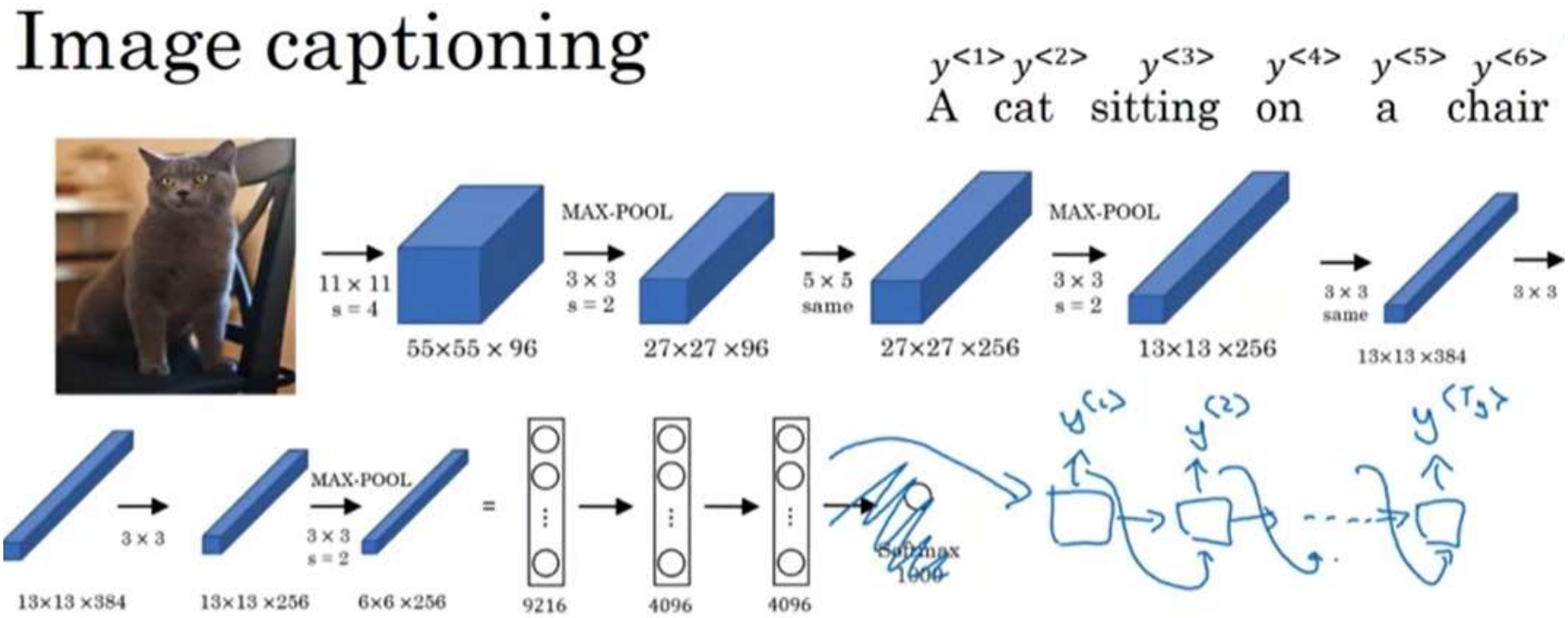
Many-to-many sequence model :

Encoder (LSTM or GRU) – X is the sentence in one language

Decoder (also RNN) – Y is the translated sentence



Sequence to Sequence architectures

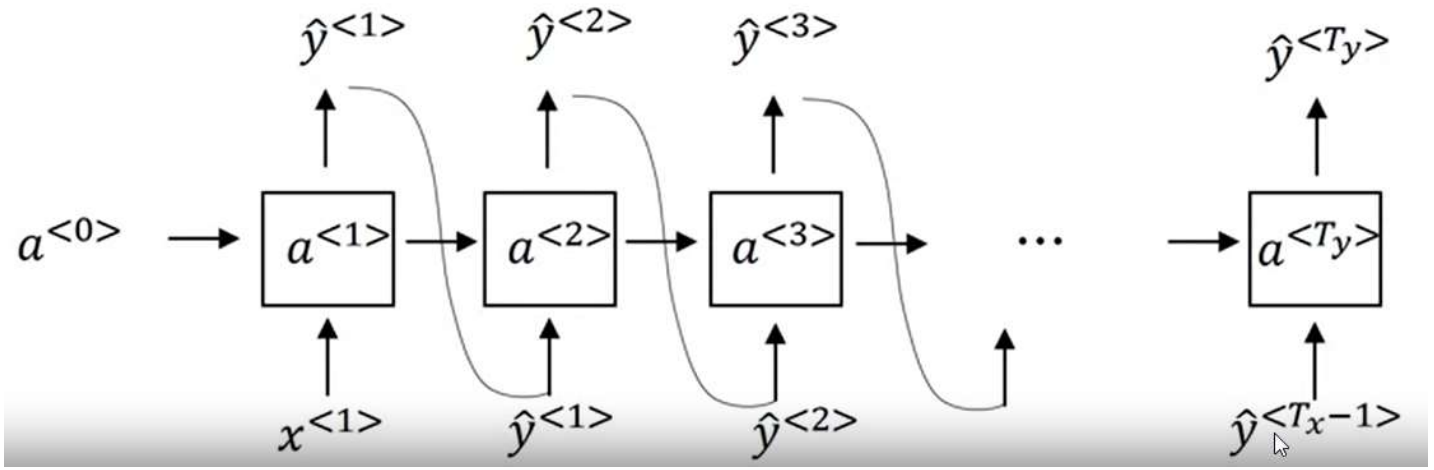


X - the image (fixed size) ;
Y – sentence that describes the image, the caption (variable size);
Pretrained CNN (e.g. AlexNet) is used as the image encoder (from image X to some representation with e.g. 4096 feature vector)
Take out the softmax end layer.
Add a RNN as the decoder to output the caption text.

Language Model vs Machine Translation

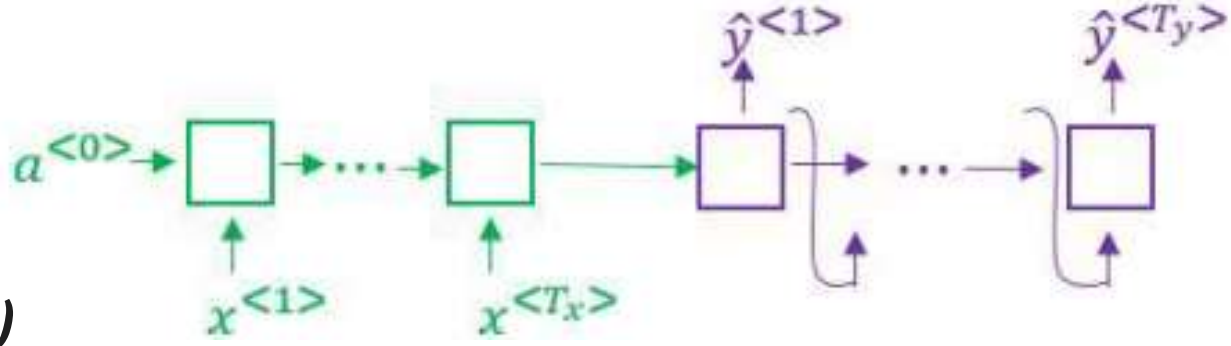
Language Model:

P(y<1> .... y<Ty>)

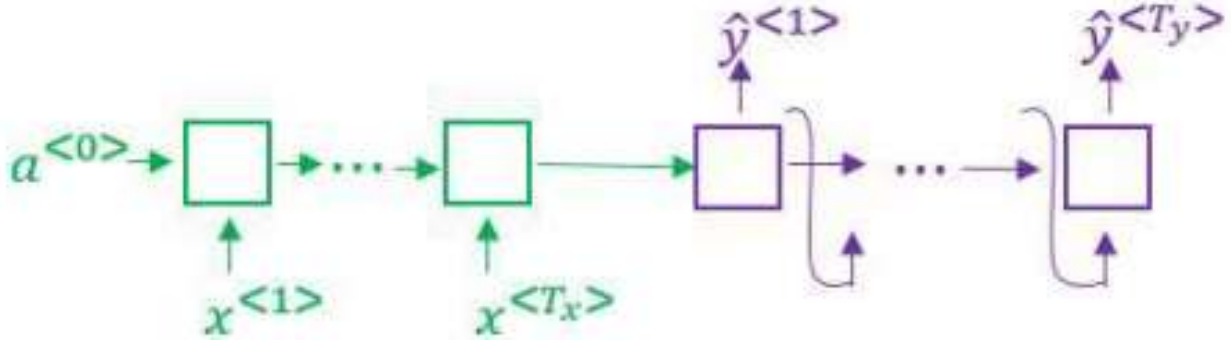


Machine translation:
Equivalent to “conditional language model“

P(y<1> .... y<Ty> | x<1> .... x<Tx>)



How to find the most likely translation



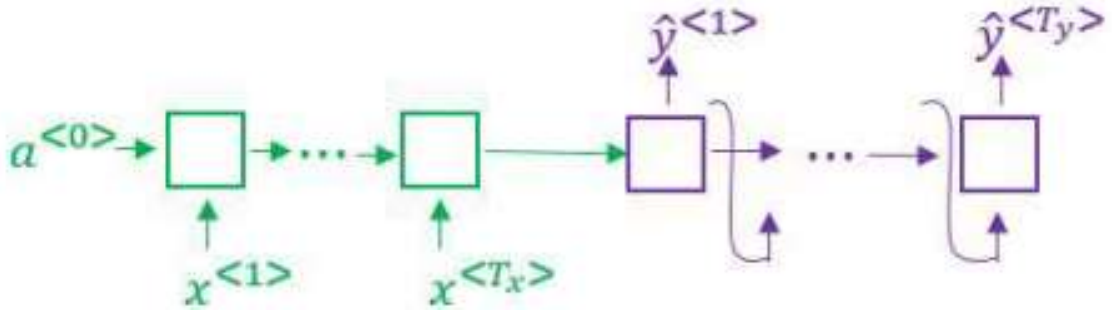
Example:

- X = "Jane visite l'Afrique en septembre."
- Y may be:
  - Jane is visiting Africa in September.
  - Jane is going to be visiting Africa in September.
  - In September, Jane will visit Africa.

So we need to get the best output it can be:

$$\arg\max_{y^{<1>},\ldots,y^{<T_y>}} P(y^{<1>},\ldots,y^{<T_y>} \mid x)$$

Algorithm 1 : greedy search



Pick the most likely first word according to the conditional language model ***P(y$^{<1>}$ | X).***
Then pick the best second word, then the best third word, and so on.

Greedy search doesn't work very well, because we want to pick the entire best sequence of words (from ***y$^{<1>}$*** up to ***y$^{<Ty>}$)*** and maximize the joint conditional probability of the whole sequence.

$$\arg\max_{y^{<1>},\ldots,y^{<Ty>}} P(y^{<1>},\ldots,y^{<Ty>}\mid x)$$

Algorithm 2: Beam Search (BS)

BS is the most widely used approximate algorithm to get the best output sequence. It has a parameter ***B*** which is the beam width. ***B=3*** means the algorithm will get 3 candidate outputs at a time.

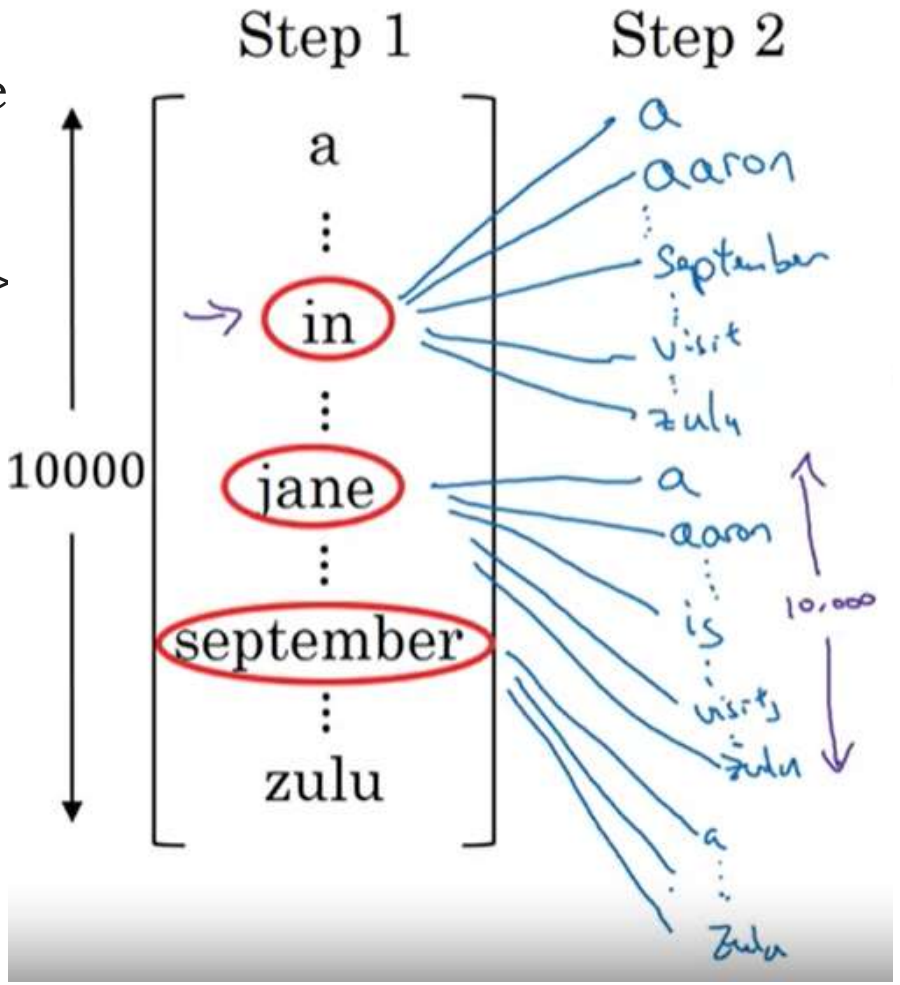
Ex. Y=Jane visits Africa in September. <EOS>

Step 1: ***B*** best candidates $P(y^{<1>} | X)$.
(e.g. we get, “in”, “jane”, “september”)

Step 2: ***B*** best combinations of 2 words

$P(y^{<1>}, y^{<2>} | X) = P(y^{<1>} | X) * P(y^{<2>} | X, y^{<1>})$

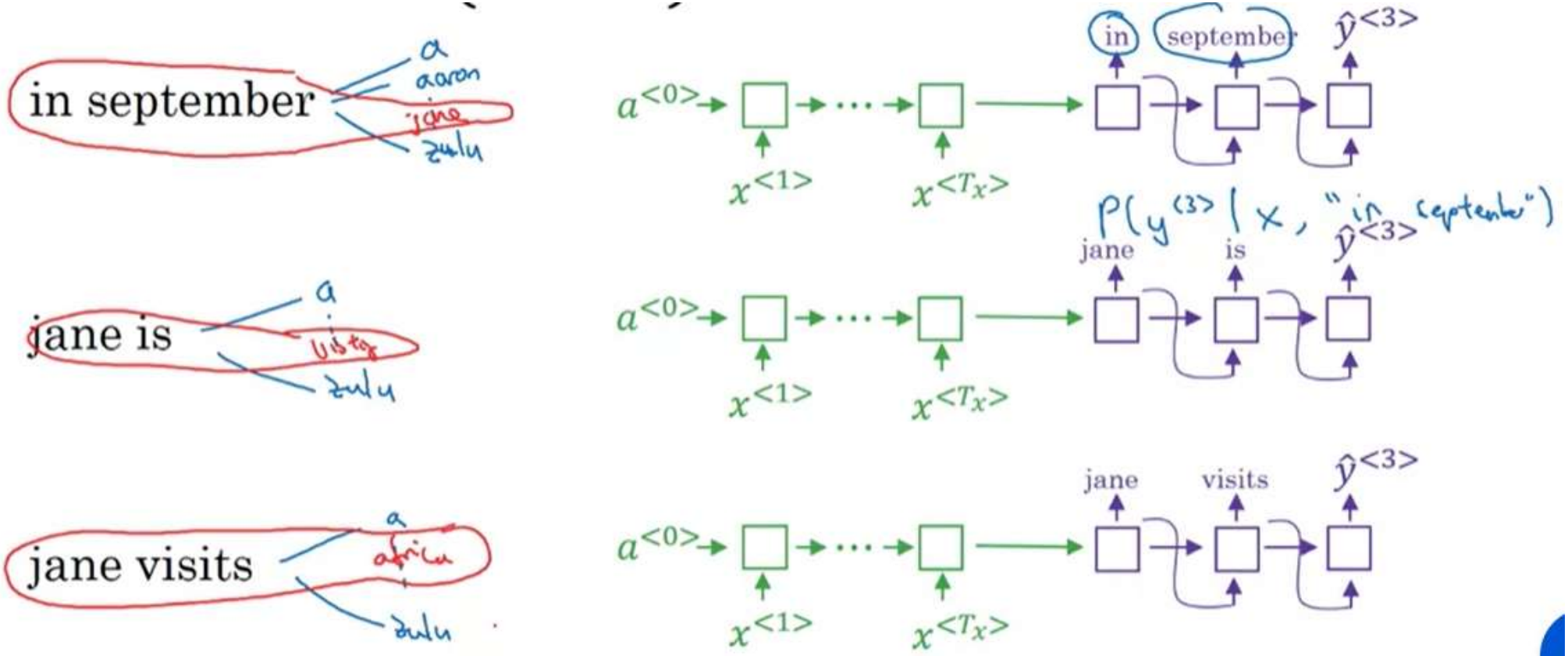
(e.g. we get, “in september”, “jane is”, “jane visit”)



Algorithm 2: Beam Search (BS)

Step 3: *B* best combinations of 3 words : $P(y^{<1>}, y^{<2>}, y^{<3>} | X)$
(e.g. we get, “in September jane”, “jane is visiting”, “jane visits africa”)
…….
and so on until the algorithm proposes as the best word <EOS>
(End Of Sentence)

If B =1 => greedy search
Larger B, better results but computationally expensive => in practice B=10



More Machine Translation terms

Beam Search is usually better than greedy search but is not guaranteed to find the exact maximum for P(Y|X).

Other search algorithms borrowed from graph theory. Similar to tree search, but the graph structure may contain cycles, so it may come to the same node again.

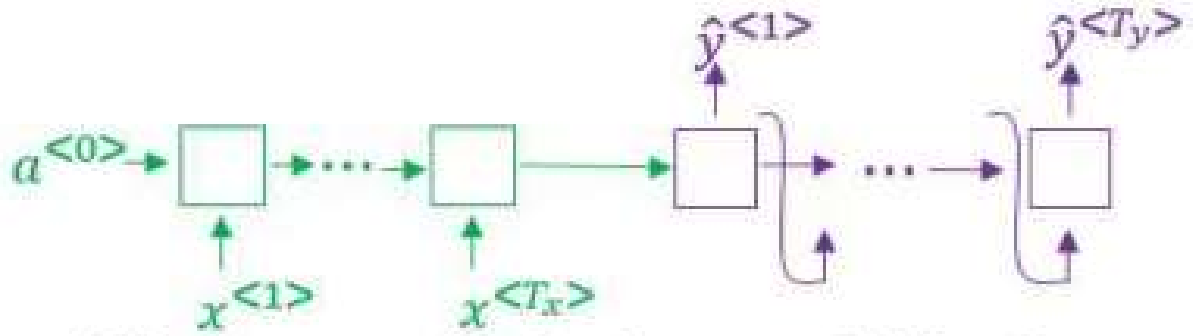
- Breadth First Search (BFS)
- Depth First Search (DFS)

Some terms:

n-grams – n words
n-gram of size 1 => “unigram”
n-gram of size 2 => “bigram”
n-gram of size 3 => “trigram”

BLEU (bilingual evaluation understudy) score (value between [0,1]**):**
it is a way to evaluate how good is the translation.
As long as the machine –generated translation is pretty close to any of the references provided by humans, then it will get a high BLUE score.

Neural Machine Translation with Attention



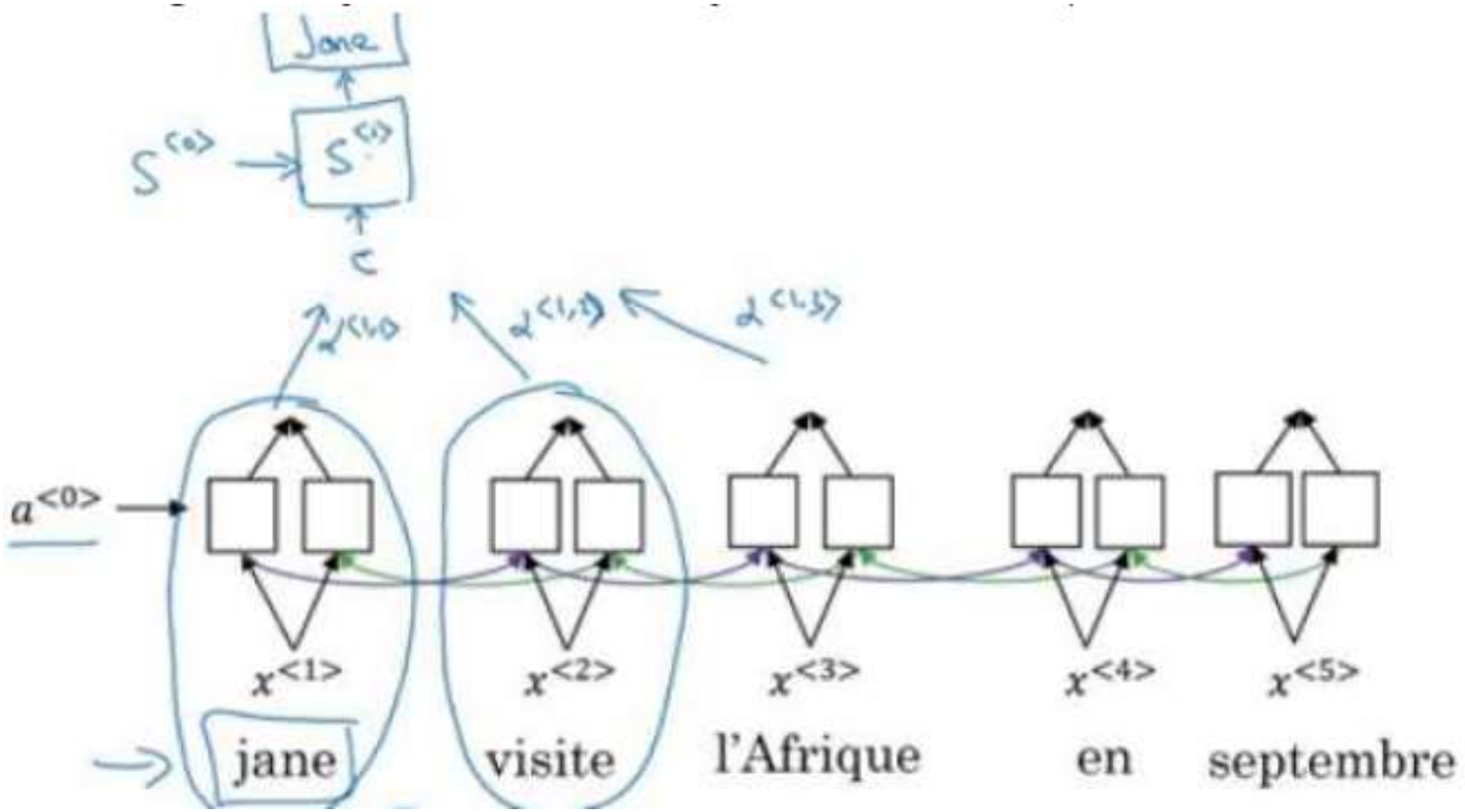
Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

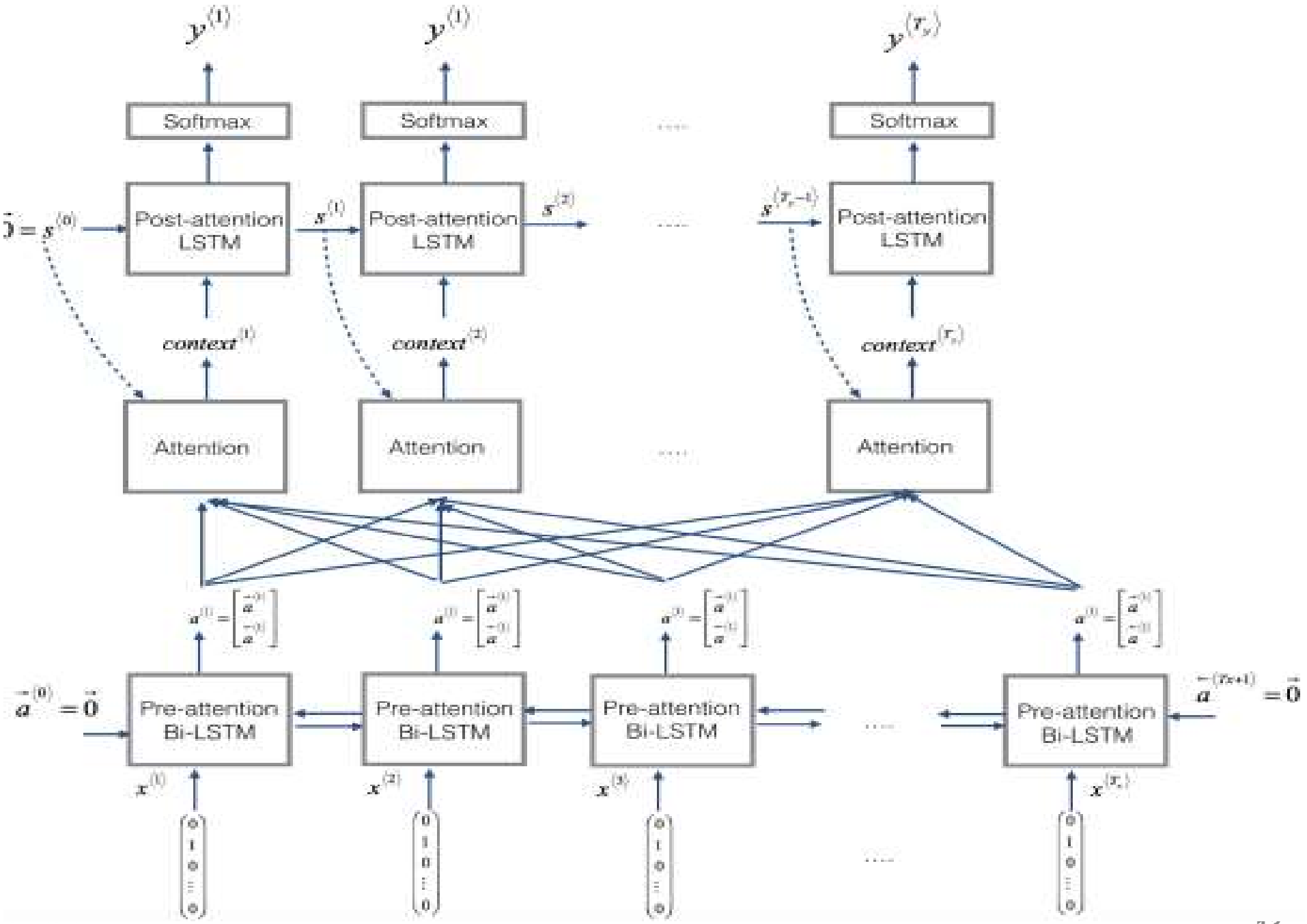
To translate a long paragraph from French to English, we often don't read the whole paragraph, and then translate.
We would read/focus on some parts of the French paragraph corresponding to parts of the English we are writing down.
The **attention mechanism** tells a Neural Machine Translation model where it should pay attention at any step.

Neural Machine Translation with Attention

Suppose the encoder is a bidirectional RNN (e.g. LSTM).
The encoder generates a vector of features that represent the French text.
The decoder is another RNN (LSTM) .
Attention weights *alfa* are used to specify which words to look with attention to generate the translated word at each step of translation.
For example to generate “jane” we will look at “jane”, “visite”, “lAfrique”.



Attention Model Architecture

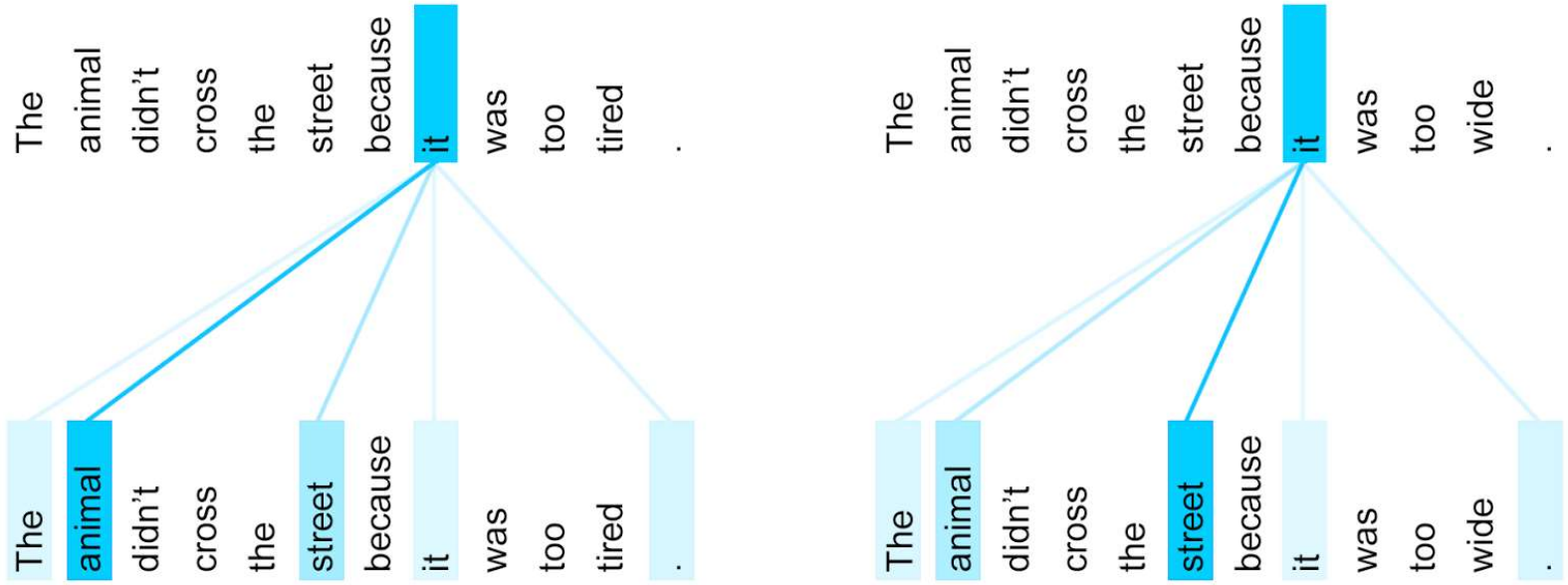


Attention Model Details

Concatenation of hidden states from the forward and backward pre-attention LSTMs

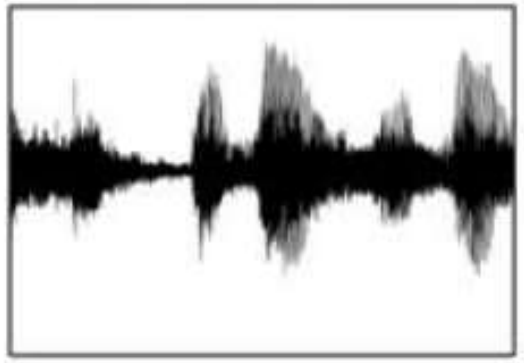
- $\overrightarrow{a}^{\langle t \rangle}$: hidden state of the forward-direction, pre-attention LSTM.
- $\overleftarrow{a}^{\langle t \rangle}$: hidden state of the backward-direction, pre-attention LSTM.
- $a^{\langle t \rangle} = [\overrightarrow{a}^{\langle t \rangle}, \overleftarrow{a}^{\langle t \rangle}]$: the concatenation of the activations of both the forward-direction $\overrightarrow{a}^{\langle t \rangle}$ and backward-directions $\overleftarrow{a}^{\langle t \rangle}$ of the pre-attention Bi-LSTM.

Block “Attention” is realized as a small NN to compute attention weights & context

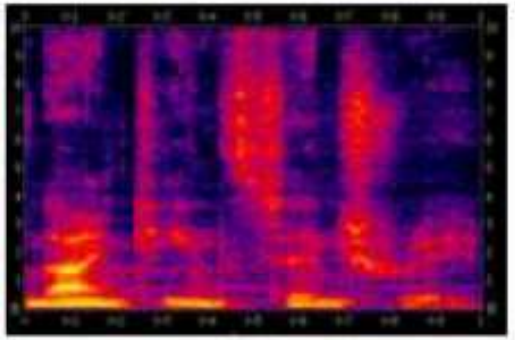


Speech Recognition – audio data

It is difficult to work with “raw” representation of audio data,
e.g. 10 sec. of audio sampled at 44100Hz = 441000 numbers.



Common pre-processing step for audio –
generate a **spectrogram =>**



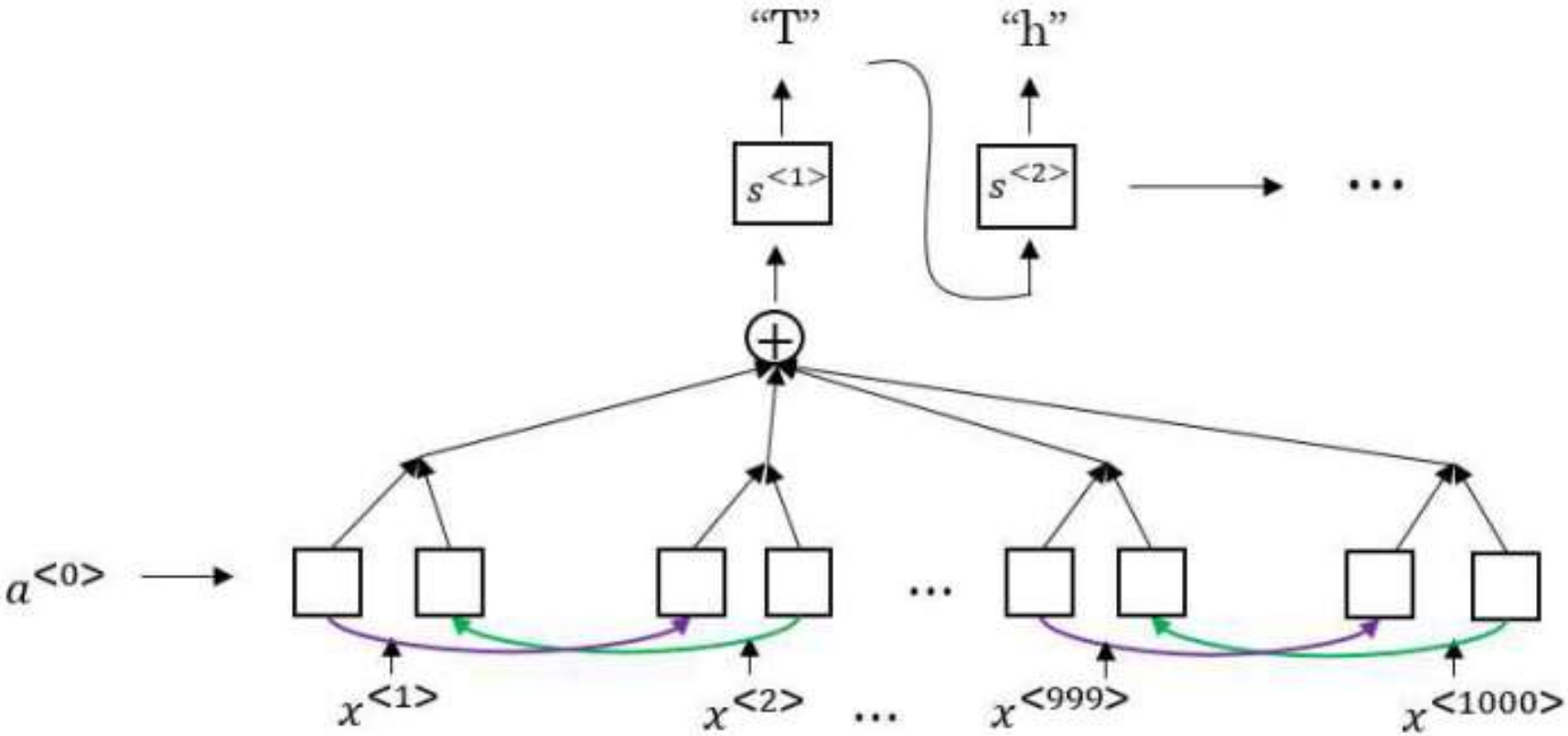
Horizontal axis is time, vertical axis frequencies.

Intensity of different colours show the amount of energy, how load is the sound for different frequencies.

Spectrogram is computed by sliding a window over the raw audio signal, and compute the most active frequencies in each window using **Fourier transformation.**

In the past, linguists were using *phonemes*.
End-to-end deep learning do not need phonemes.

General Attention model for Speech Recognition

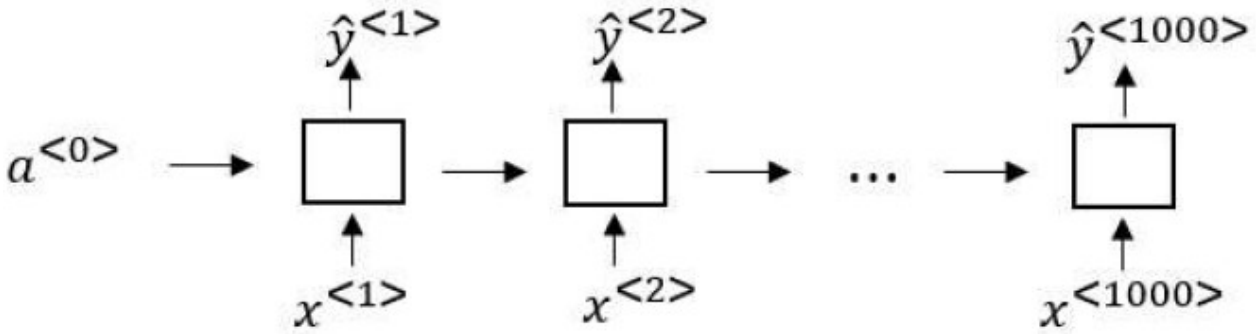


Attention model:

Input of **encoder**: different time frames of the audio clip X
Bidirectional RNN, extracts features from the sequence of audio frames

Output of **decoder** : transcript Y (e.g. "The quick brown fox")

CTC cost for Speech Recognition



CTC – Connectionist Temporal Classification

RNN (Uni or Bidirectional) with the same number of inputs and outputs, but in speech recognition input X tends to be a lot larger than output Y.

E.g. 10 sec. of audio at 100Hz gives X with shape (1000,1), but the output will not have 1000 characters.
The CTC cost function allows RNN to output something like this:

ttt_h_eee<SPC>___<SPC>qqq___

_ is a special character called “blank” and <SPC> is for “space” character.

Basic CTC rule: collapse repeated characters not separated by “blank” .
19 characters in Y= “The quick brown fox” can be generated by 1000 character output using CTC and it´s special blanks.
Using **both attention model and CTC** can build an accurate speech recognition system (Baidu’s DeepSpeech) .

Trigger Word Detection System

Amazon Echo
(Alexa)

Baidu DuerOS
(xiaodunihao)

Apple Siri
(Hey Siri)

Google Home
(Okay Google)

Wake up a device by saying some trigger words.

Still evolving topic, not a single agreed algorithm.

RNN – disadvantages

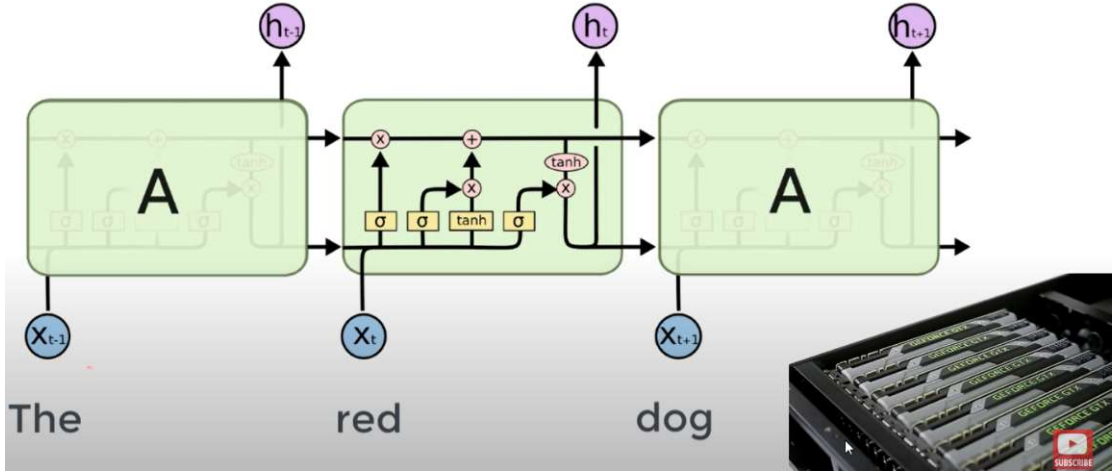
- RNN are slow to train
- Long sequences => lead to vanishing/exploding gradients

- LSTM/GRU – some improvements to long sequence training but even slower to train than RNN.

Ex. The most likely meaning of the word "bank" in the sentence "I arrived at the bank after crossing the…" requires knowing if the sentence ends in "… road." or "… river."

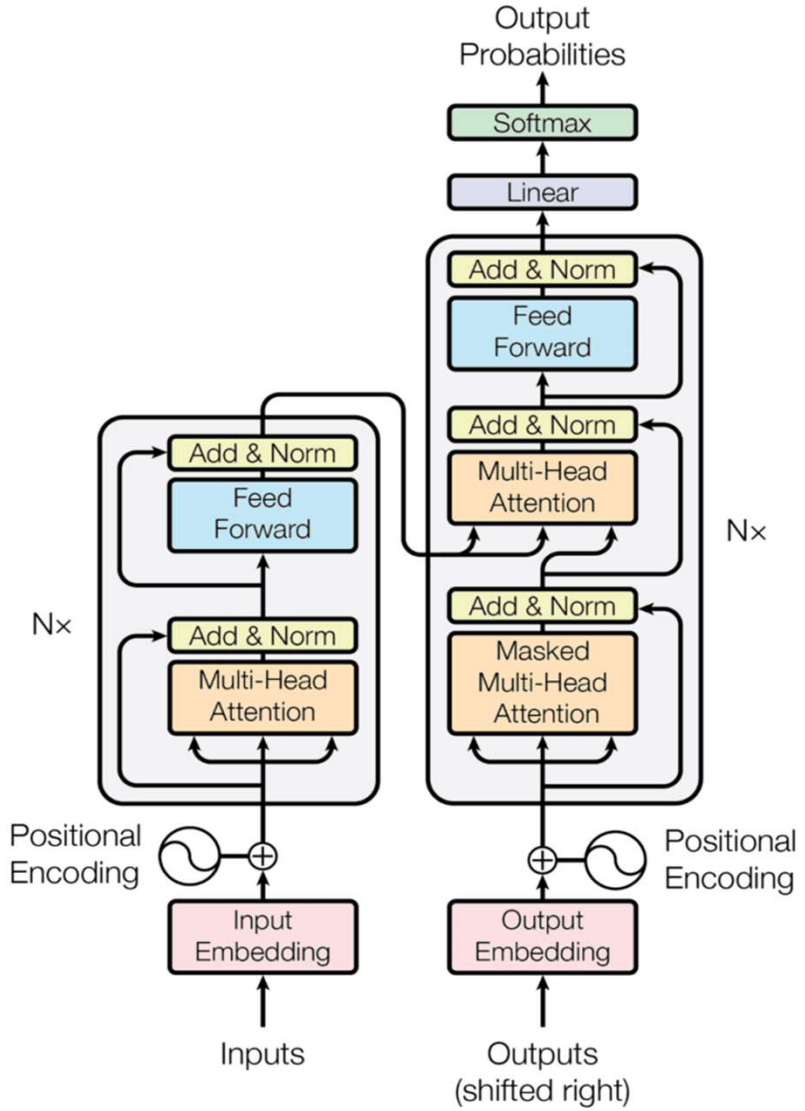
- Input data is passes sequentially (one word/token at a time).
Need inputs from previous state to make operation on the current state.
RNNs do not make good use of GPUs designed for parallel computation.

How can parallelize sequential data ? => **Transformers (2017)**



Transformer architecture (Attention)

It follows an encoder-decoder structure but does not rely on recurrence and convolutions in order to generate an output.



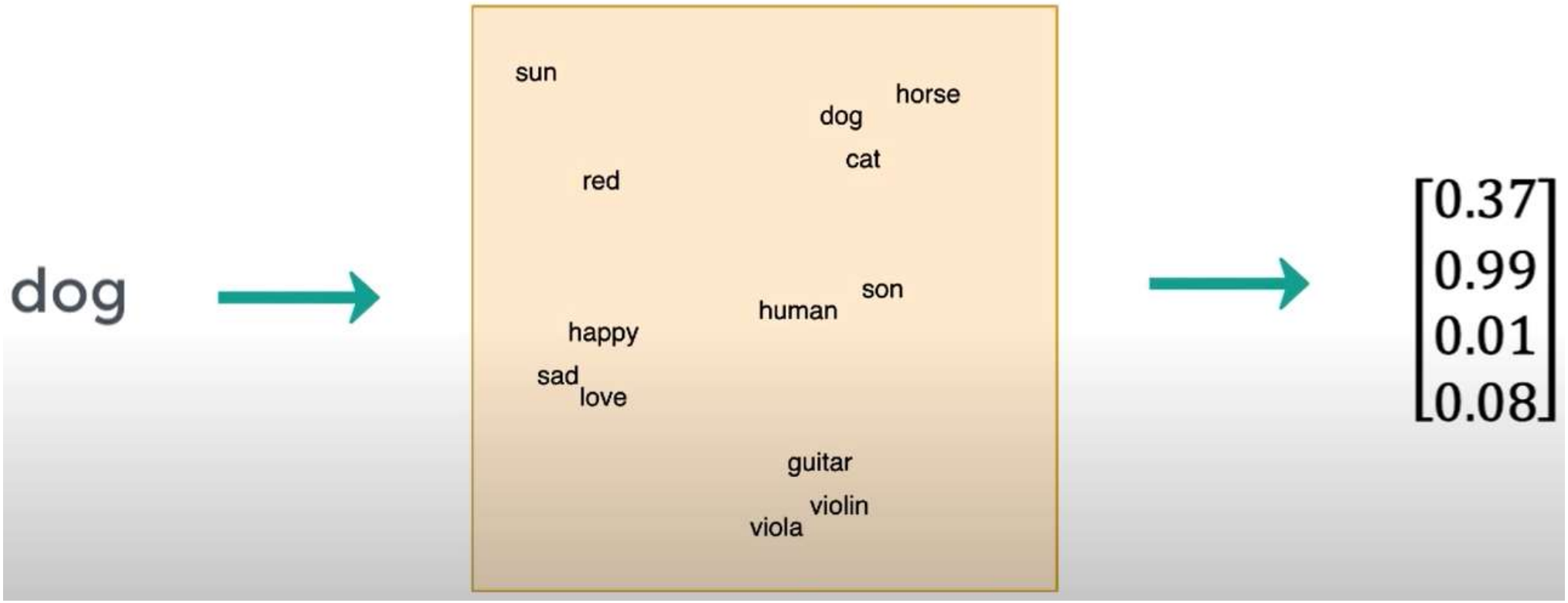
Ashish Vaswani et al., Attention Is All You Need, 2017

Encoder (left half) maps an input sequence to a representation, which is then fed into a decoder.

Decoder (right half) receives the output of the encoder together with the decoder output at the previous time step to generate an output sequence.

Input Embedding

Embedding space : map every word to a point in space where similar words in meaning are physically closer to each other.
Embedding space can be pretrained to save time or use already pretrained, e.g. Glove. Here we assume embedding space = 4.



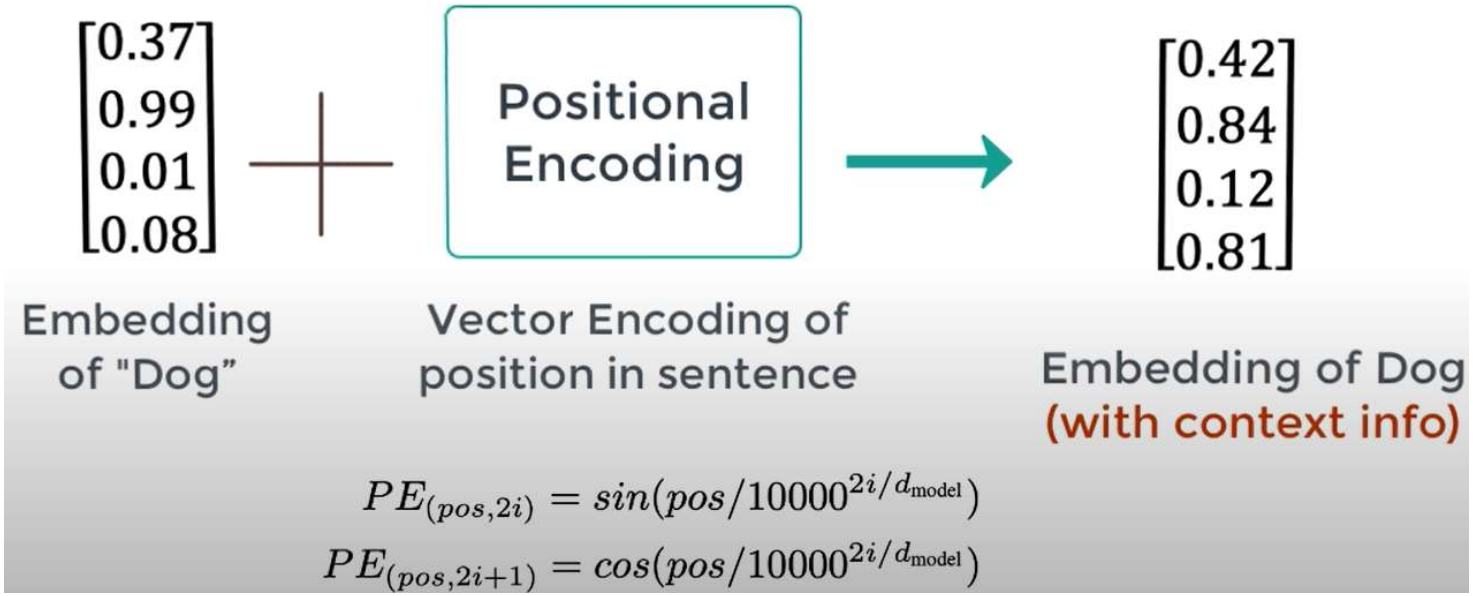
Positional Encoding

The same word in different sentences may have different meaning =>
Positional Encoding (PE) gives context based on position of word in text.

His dog is cute => pos = 2;
He looks like a dog => pos = 5

pos – numerical position of the word.
i – index in the embedding vector
d – dimension of the embedding vector (d=4)

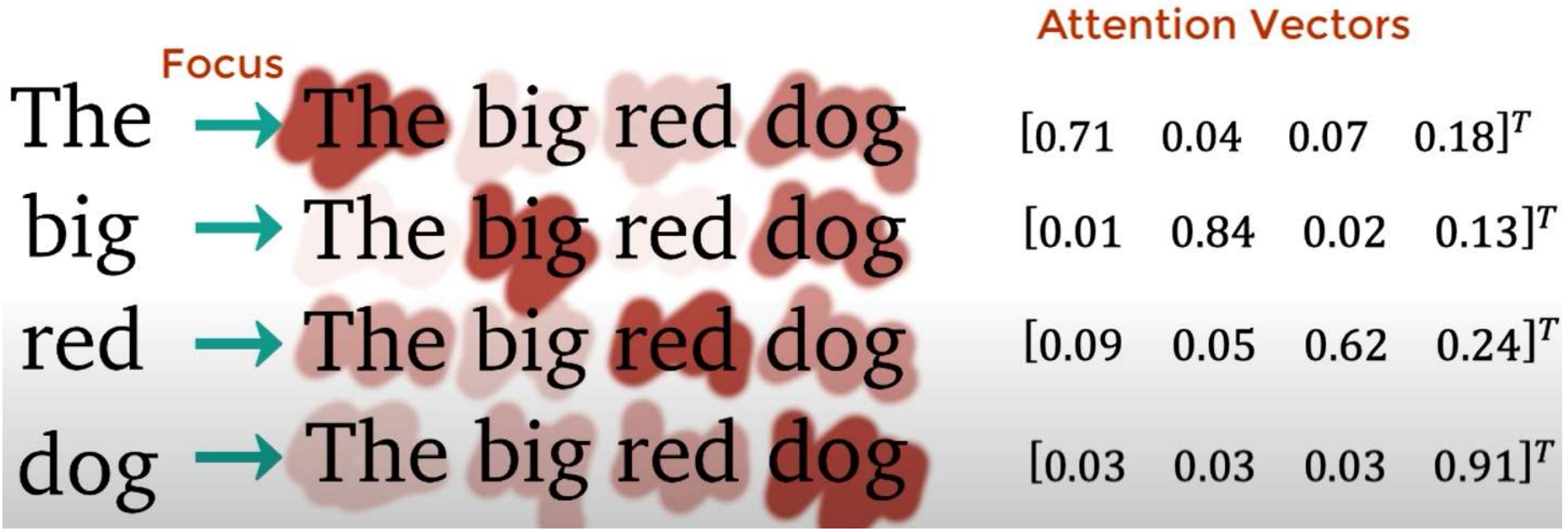
Original paper uses sin & cos but it can be other function.



Attention

What part of the input should we focus ?
How relevant is one word to the other words in this sentence ?
Several attention vectors.

Encoder:



Transformers way to compute Attention

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k^{<i>})}{\sum_j \exp(q \cdot k^{<j>})} v^{<i>}$$

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$

Every word, is associated with 3 values (q- the query; k-key; v- value pairs).
$X^{<i>}$ - word embedding of *i* word =>

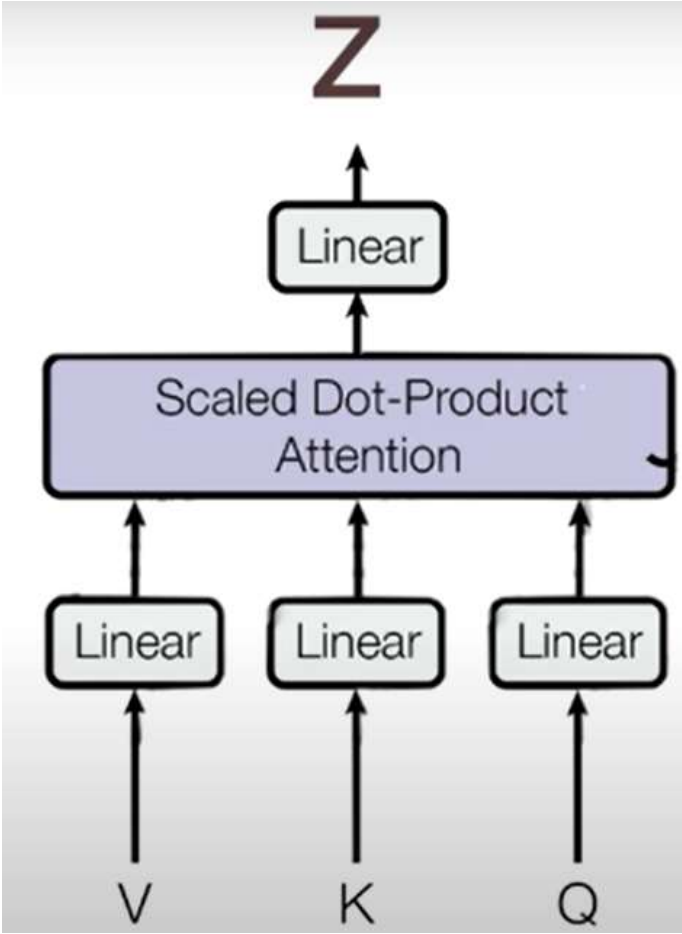
$q^{<i>} = W^Q X^{<i>}$, $k^{<i>} = W^k X^{<i>}$; $v^{<1>} = W^v X^{<i>}$;

W^Q , W^k , W^v – parameters to be learned.

Loose analogy to database concept where we have queries and key-value pairs. $q^{<i>}$ is a question about the *i* word, e.g. what's the subject?

The goal of Attention is to compute more useful/rich representation (features) $A^{<i>}$ for each word.

Attention (Head)



$$Z = softmax\left(\frac{Q.K^T}{\sqrt{Dimension\ of\ vector\ Q, K\ or\ V}}\right).V$$

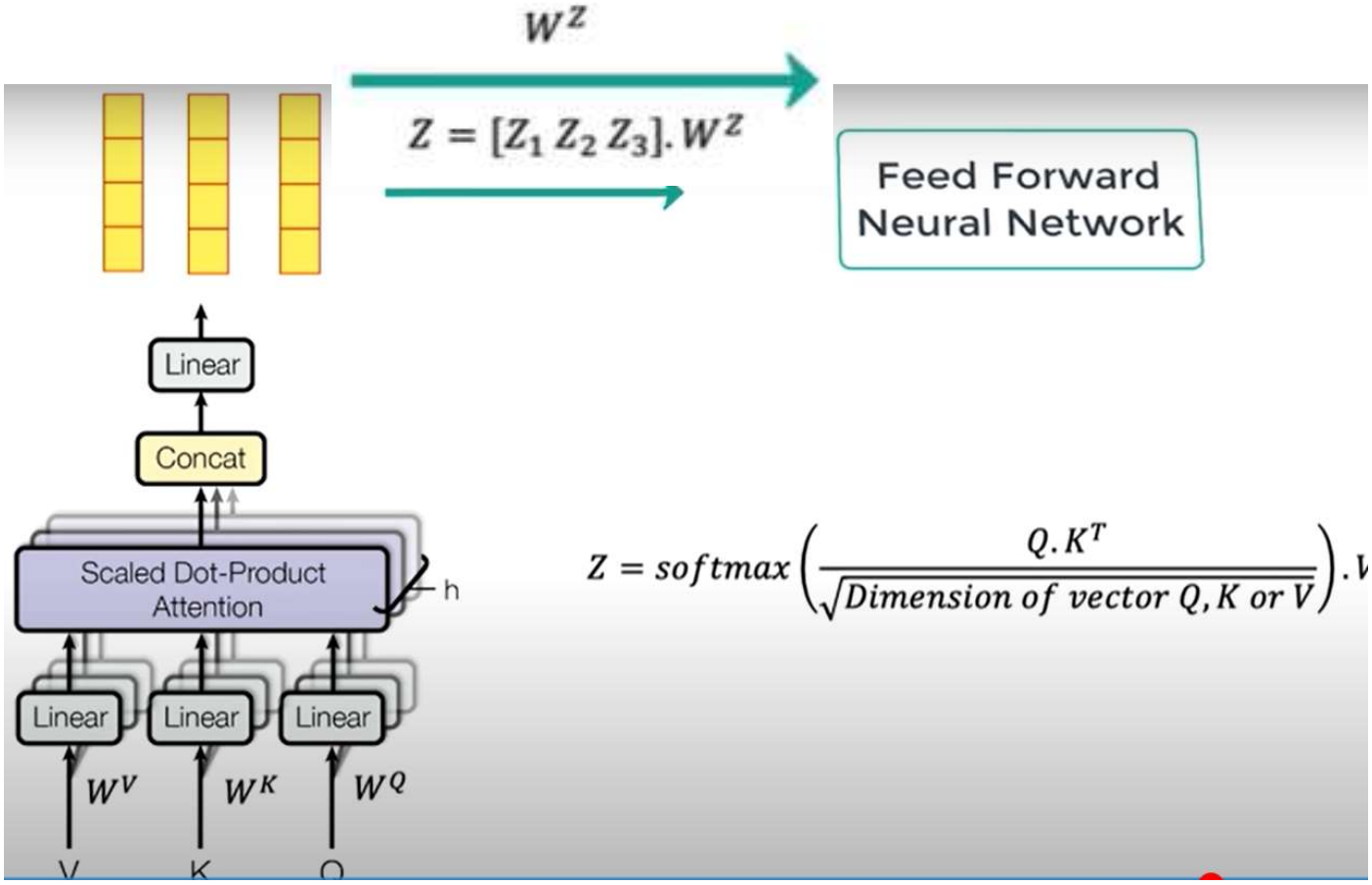
Multi-head Attention

Create several Attention heads (h).

Intuition: each Attention will be an answer to a different query, e.g. what's the subject? what happens ? when ? where ?, etc.

In all these representations (Z), different words will have the greatest weights (importance). They have the meaning of rich features and are concatenated.

Z is then passed to a Feed Forward NN.

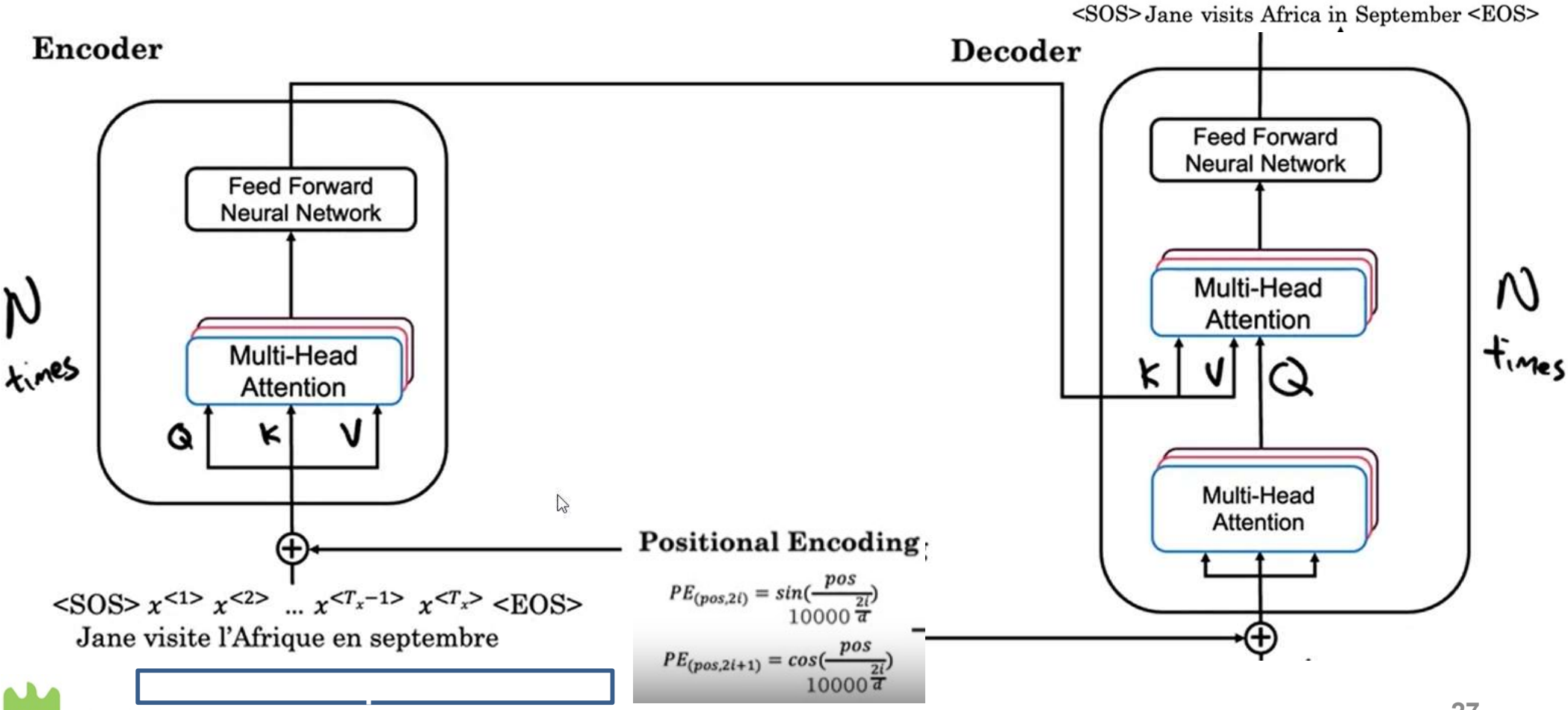


Transformer Encoder-Decoder

The output of the Encoder is feed to the Decoder block.

1st Multi-Head Attention block at the decoder inputs the already generated words of translation.

2nd Multi-Head Attention block at the decoder inputs information (query) from already translated words and K, V represent the context from the original text and try to decide what is the next word of translation.



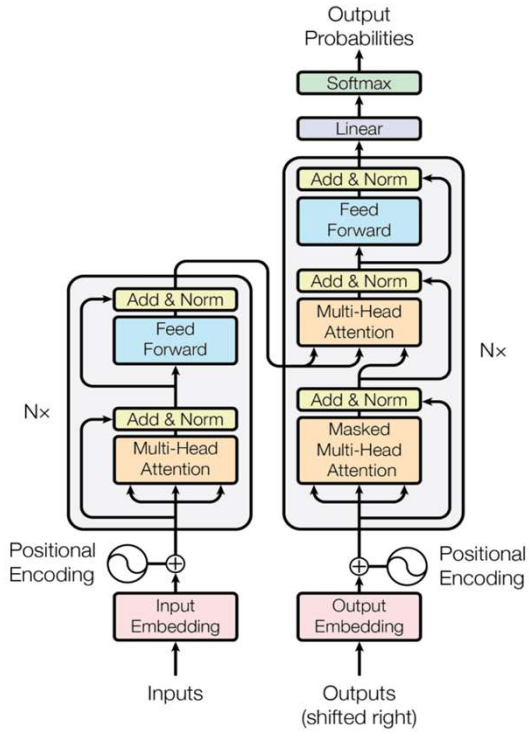
Attention at the Decoder block

How relevant is one word to the other words in the translated sentence ?

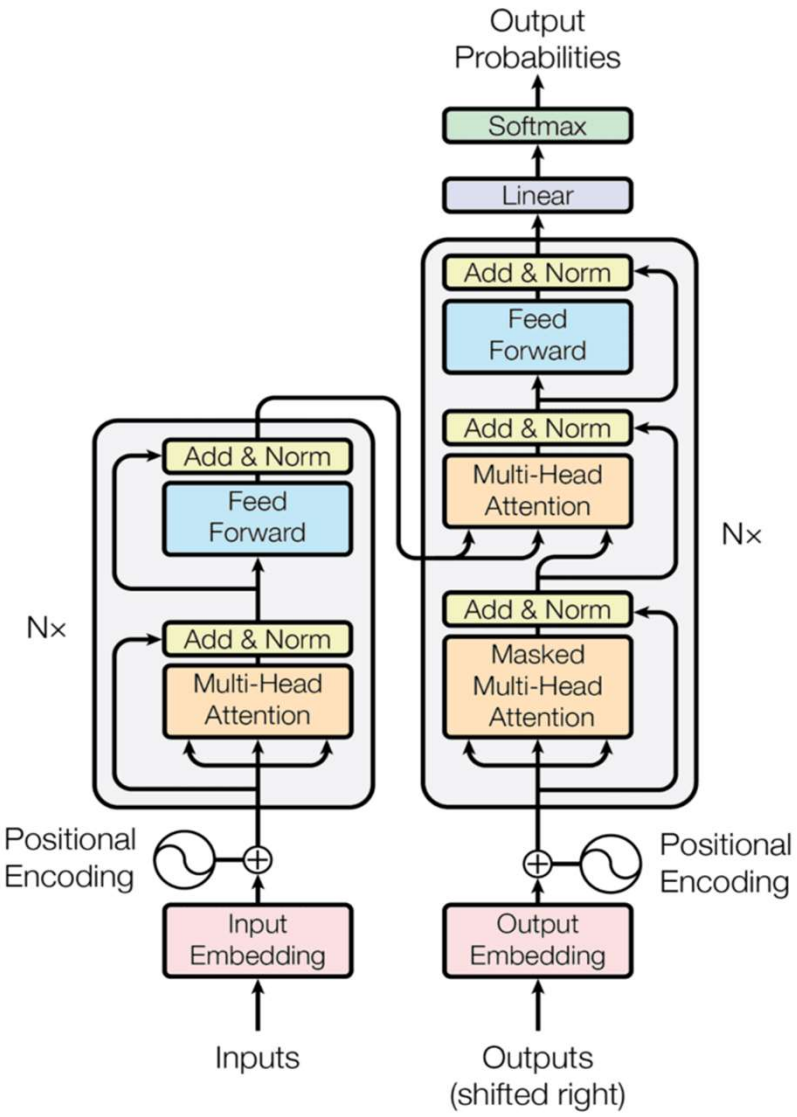
Encoder-Decoder Attention

1st Multi-Head Attention block is masked during the training, i.e. it blocks the last part of the sentence to mimic what the network will do during test/prediction time.

2nd Multi-Head Attention block at the decoder mixes encoder-decoder attention vectors.



Transformer



The output of the embedding block contains contextual, semantic embedding and positional encoding information.

The output of the embedding layer is a matrix with dimension:

d (embedding vector) x (max length of sequence the model can take)

The outputs of all other layers have the same shape.

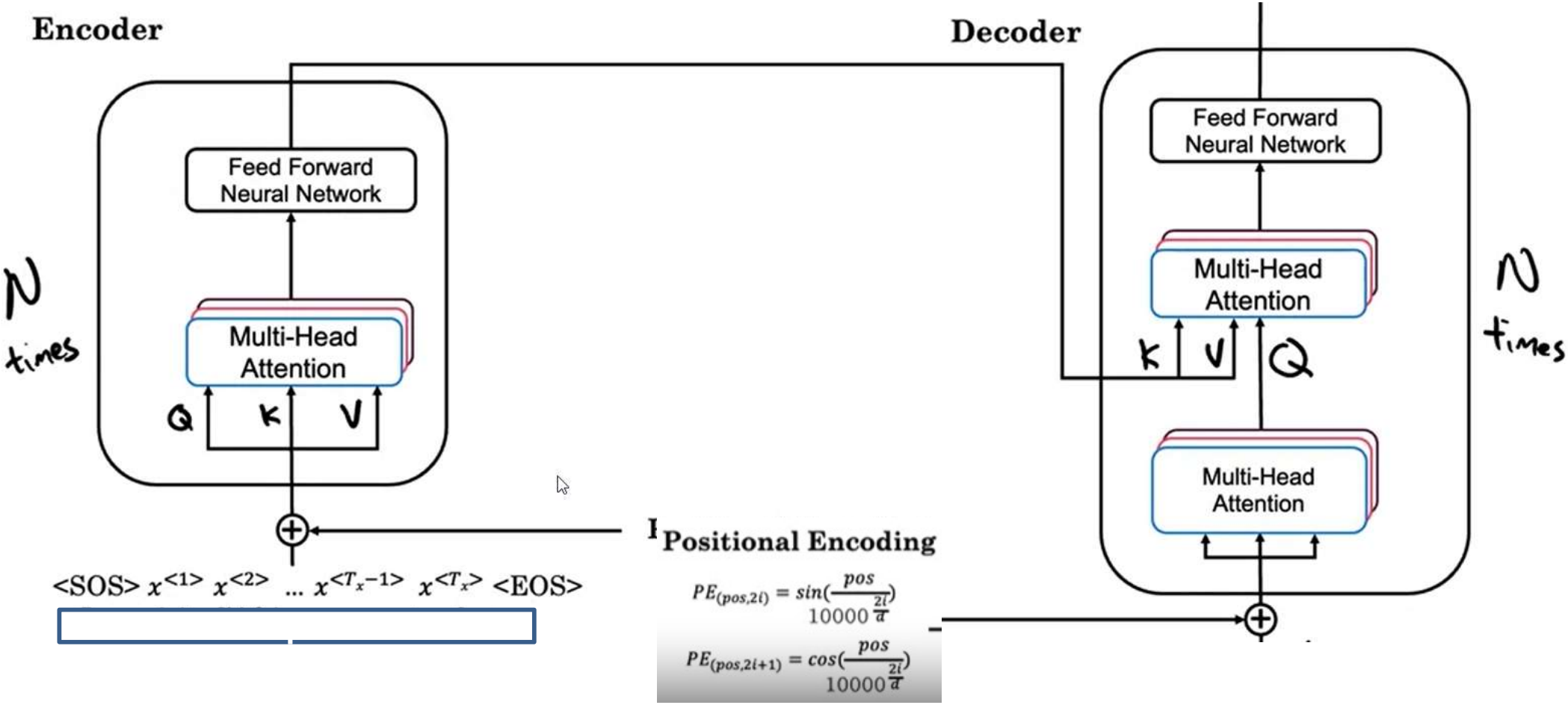
The output pass trough FF networks with residual connections (as with the ResNet).

Add & Norm layers are repeated (similar to Batch Normalization), they help speed up learning.

Softmax layer predicts the next translated word (one word at a time).

Transformer encoder-decoder

Not only for sequence-to-sequence translation task.
Also, for conversation:
- Encoder input: How are you ?
- Decoder output: I am fine.



Generative Pretrained Transformer (GTP) models

Transformers: BERT, GPT-2, RoBERTa T5 - top performance on a wide array of language tasks. Applying unsupervised, self-supervised learning, or learning without human-labeled data.

GPT-3: language used to instruct a large NN to perform a variety of text generation tasks.

Image GPT: the same type of NN can also be used to generate images from text.

DALL-E (Zero-Shot Text-to-Image Generation) :
12-billion parameter version of GPT-3 trained to generate images from text descriptions, using a dataset of 250 million text-images pairs from the internet. DALL-E combines NLP techniques with computer vision.

Energy/storage/computational issues

Modern AI architectures face those problems : energy consumption, storage space, computational recourses.

These issues are open research problems with great interest and dynamics.

Transformer (65 million parameters): needs 12h for training, 1 server with 8 Nvidia Tesla P1001. Consume 27kW/hour, produce 1 CO2 lbs (pounds).

BERT (language models based on Transformers): needs 79h for training, with 64 Nvidia Tesla V100. Consume 1.507 kWh, produce 1.438 CO2 lbs.

DALL-E (12-billion parameters): needs 40-45 days for training on TPUs (Tensor Processing Units) v3-256. Needs about 24 GB of memory when stored in 16-bit precision, which exceeds the memory of a 16 GB NVIDIA V100 GPU.

ChatGPT (175 billions parameters) created to produce text similar to a human, needs 800 GB memory.

Industrial Revolutions

Industry 1.0 (1760) - Invention of the steam power & steam engine

Industry 2.0 (1871-1914) - Invention of electricity; railroad, telegraph networks

Industry 3.0 (1970) – invention of computer, digital revolution

Industry 4.0 (2011) – data-driven approach to automate processes, IoT, AI

Industry 5.0 (now) – people co-working with smart machines/robots

Industry 6.0 (a futuristic idea) - Hyper-connect industries to industries, customer-driven, highly mass personalization of services and products. Information/data flow all across the countries.