Multi-head Attention and Transformer

Tirtharaj Dash

Dept. of CS & IS and APPCAIR
BITS Pilani, Goa Campus

November 20, 2021

In the last lecture:

(1) attention – mechanism.

(2) self-attention. "block"

(Q , K , V)

↓

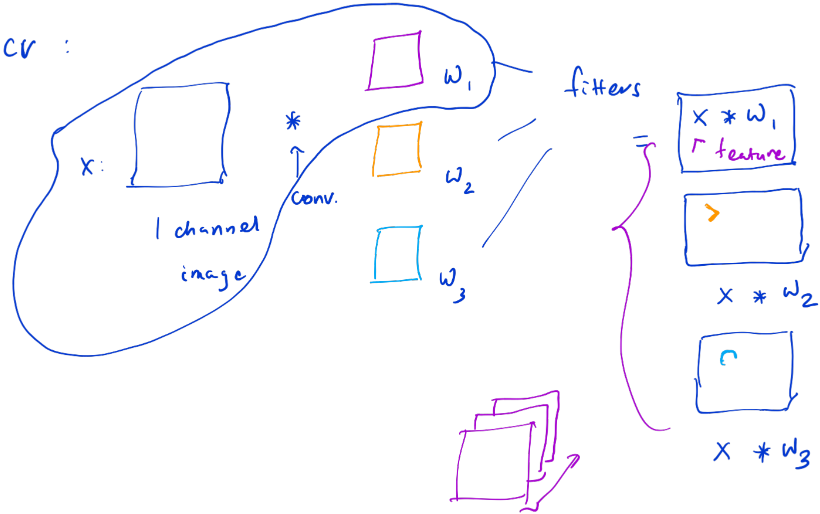
Next:

Multiheaded attention.

"multi" "head"

" multiple self-attention "

Understanding / Intuition of multihead attention :-



Self-attention :-

Multiple self-attention.

"Multihead attention"

Sequence: "$\boxed{I}$ want to live in $\boxed{\text{Europe.}}$"

(Q, K, V)

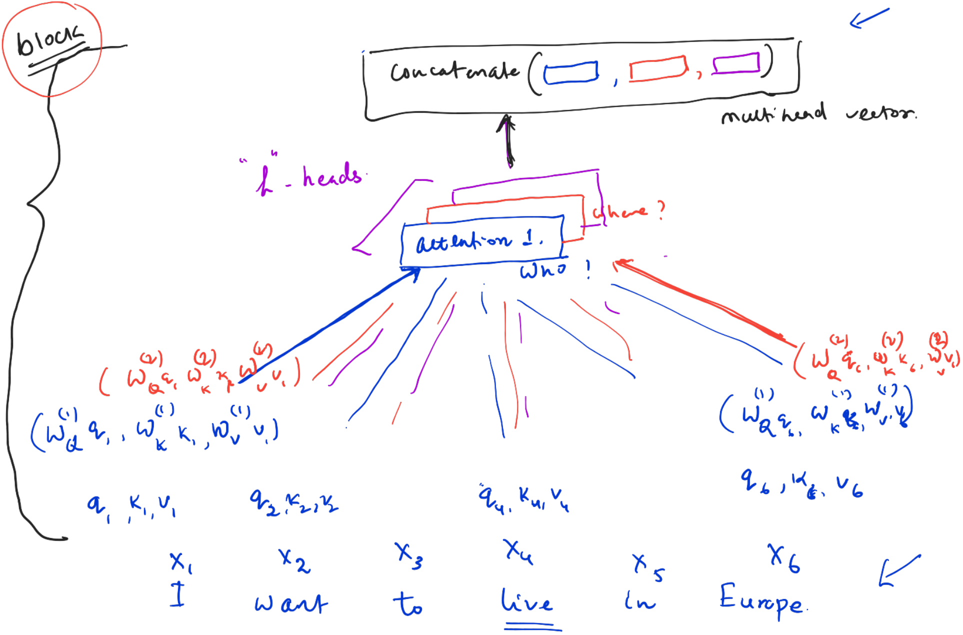
self-attention

"head 2" $\left(W_Q^{(2)}, W_K^{(2)}, W_V^{(2)}\right)$

$\left(W_Q^{(1)}, W_K^{(1)}, W_V^{(1)}\right)$ first feature "head 1"

$\boxed{\text{Where ?}}$

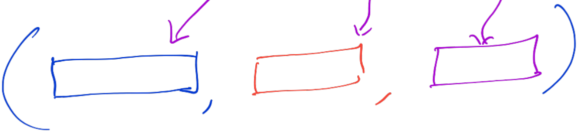
feature: $\boxed{\text{"Who ?"}}$


block
Concatenate (, ,)
multi head vector.
"h"-heads.
attention 1.
where ?
Who !
$(W_Q^{(2)} q_1, W_K^{(2)} k_1, W_V^{(2)} v_1)$
$(W_Q^{(1)} q_1, W_K^{(1)} k_1, W_V^{(1)} v_1)$
$(W_Q^{(2)} q_6, W_K^{(2)} k_6, W_V^{(2)} v_6)$
$(W_Q^{(1)} q_6, W_K^{(1)} k_6, W_V^{(1)} v_6)$
q_1, k_1, v_1
q_2, k_2, v_2
q_4, k_4, v_4
q_6, k_6, v_6
x_1 x_2 x_3 x_4 x_5 x_6
I want to live in Europe.

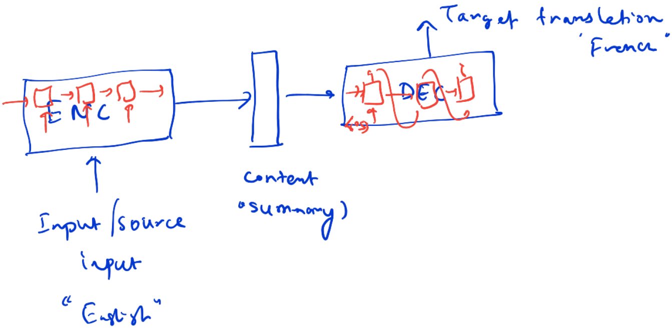
$$\text{attention score}\,(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

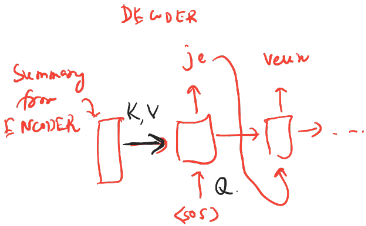
→ Self-Attention allows us construct "rich" embeddings representations for the input sequence

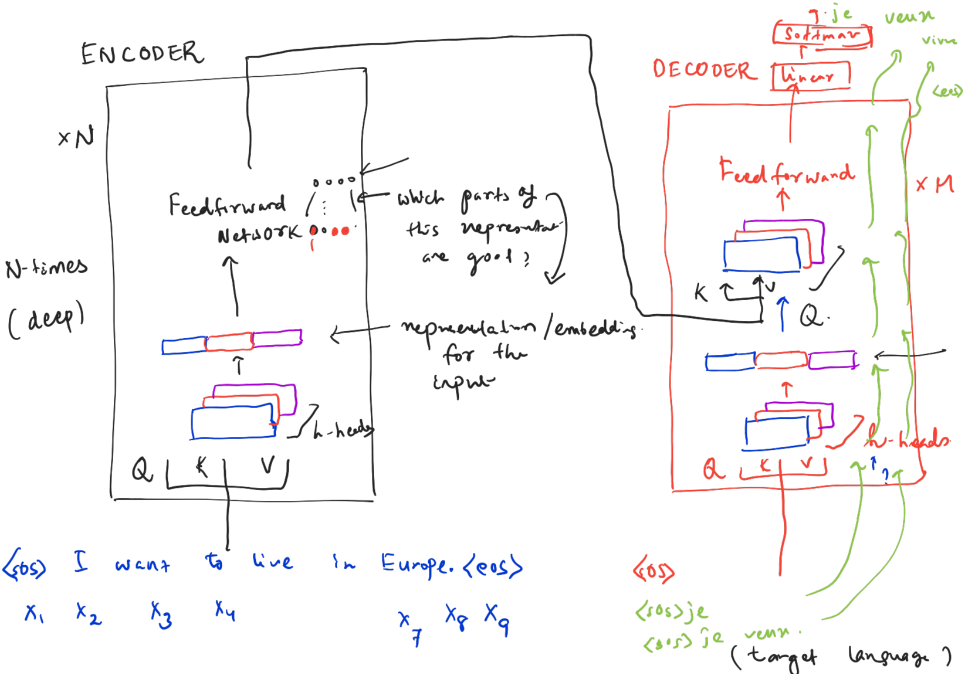
→ Multihead attention



Transformers:- Seq2seq (Machine translation)






ENCODER
x N
N-times
(deep)
Feedforward Network
which parts of this representation are good?
representation / embedding for the input
h-heads
Q K V
<sos> I want to live in Europe. <eos>
x_1 x_2 x_3 x_4 x_7 x_8 x_9
DECODER
Softmax
linear
je
venir
viva
<eos>
Feed forward
x M
K V Q
h-heads
Q K V
<sos>
<sos> je
<sos> je venir
(target language)

Some additional machinery in Transformer (Enc, Dec):-

(1) Model might miss the "importance" of positions of the words in the input sentence.

→ Positional encoding. (explicitly providing these info.)

(2) Stabilising training.

→ akin "Batch Norm" (CNNs, MLPs)

"add & norm" → Layer Normalisation. (sequence models)

Positional Encoding:

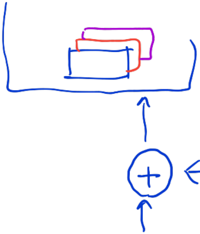
Words : word <embedding>

word_1 $\quad$ word_2 $\quad \ldots \quad$ word_{T_x}.

< > $\quad$ < > $\quad$ < >

< pos info > $\quad$ < pos info > $\quad$ < pos info >

Positional encoding using trigonometric functions
(sine, cosine)



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

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

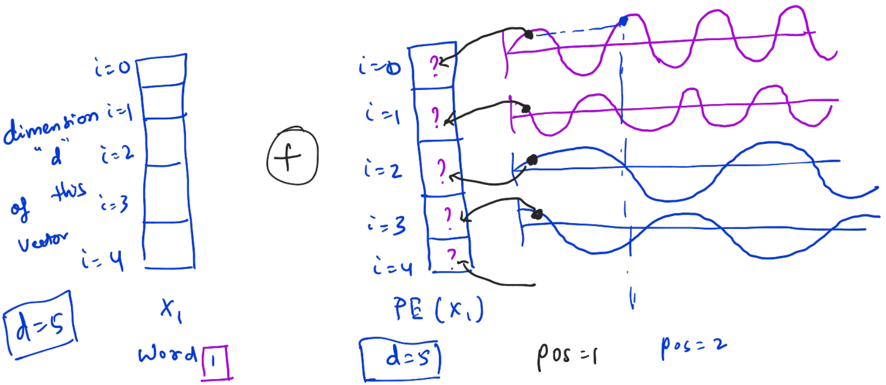
Positional encoding.

ENC
DEC

<sos> I want to live in Europe <eos>

Word embedding vectors

$$\sin\left(\frac{pos \to 1}{10000^{2i/d}}\right) \to n$$



pos=1

adding directly.

Word pos = x.

dictionary size = 10000

< 10000 dimen >

Word embedding : encoding the word

Pos. encoding : ordering or positions

$\oplus$

Why sine / cosine ?

→ range [-1, 1]

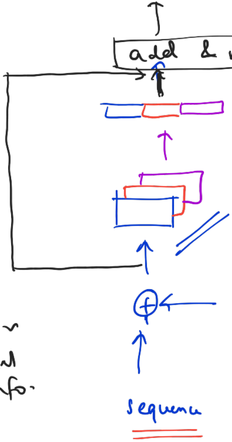
small

→ order: Periodicity in these functions

$\boxed{d\,?}$ What exactly is d ? (Max length of the sentence in the input / source language)

Add & Norm:

Skip (residual connection)



ordering on positional info.

PE of the sequence = d

same is done for the DECODER.

Homeworks:

1. Layer normalisation

   (vs. BatchNorm)

2. Decoder: Multihead attention block.

   "Masked multihead attention"

   ↑

   (Training)