Understanding Transformers: A step-by-step hand simulation to translate a sentence from English to Hindi

- How information changes through each layer of a basic transformer.
- Assumptions: weights are already optimized, so no backpropagation required

let's	1	0	0	0
to	0	1	0	0
go	0	0	1	0
<EOS>	0	0	0	1

$$\text{English vocabulary} = I_4 = \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}$$

Word embedding:

$$I_4 . \boxed{\begin{matrix} 1.87 & 0.09 \\ -1.45 & 1.50 \\ -0.78 & 0.27 \\ 2.21 & -0.64 \end{matrix}} = \begin{array}{cl} \boxed{\begin{matrix} 1.87 & 0.09 \\ -1.45 & 1.50 \\ -0.78 & 0.27 \\ 2.21 & -0.64 \end{matrix}} & \begin{matrix} \text{let's} \\ \text{to} \\ \text{go} \\ \text{<EOS>} \end{matrix} \end{array}$$

$$\text{Sentence} = [\text{let's}, \text{go}] = \begin{array}{cl} \boxed{\begin{matrix} 1.87 & 0.09 \\ -0.78 & 0.27 \end{matrix}} & \begin{matrix} \text{let's} \\ \text{go} \end{matrix} \end{array}$$

- Weights are optimised using backpropagation
- The process of optimising the weights is called training

But position matters!

[do , I , like , this]
[I , do , like , this]

different meanings, so just embedding words as vectors won't work

we need to somehow embed positions

<do, 0>; <I, 1> ; <like, 2> ; <this, 3>

<I, 0> ; <do, 1>

so that the same words could be embedded differently if their position changes their meaning

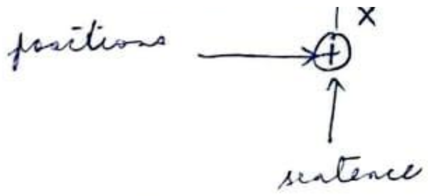
Positional embedding:

0 = <0, 1>
1 = <-0.9, 0.4>

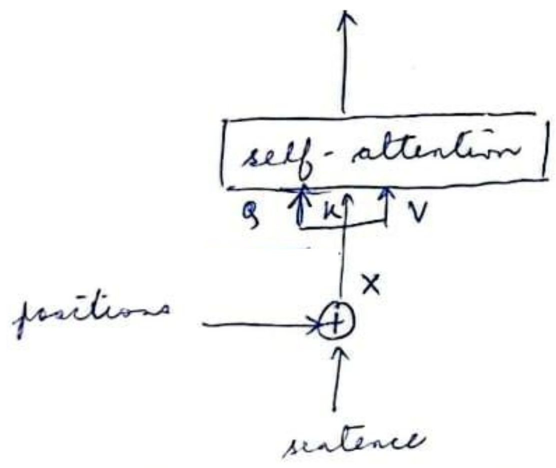
$$X = \text{sentence} + \text{embedded positions}$$

$$= \begin{array}{ll} \boxed{\begin{matrix} 1.87 & 0.09 \\ -0.78 & 0.27 \end{matrix}} & \begin{matrix} \text{let's} \\ \text{go} \end{matrix} \end{array} + \begin{array}{ll} \boxed{\begin{matrix} 0 & 1 \\ -0.9 & 0.4 \end{matrix}} & \begin{matrix} 0 \\ 1 \end{matrix} \end{array}$$

$$X = \begin{array}{ll} \boxed{\begin{matrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{matrix}} & \begin{matrix} \langle \text{let's}, 0 \rangle \\ \langle \text{go}, 1 \rangle \end{matrix} \end{array}$$



positions
x
sentence



Self-attention keeps track of the relationships among words

The stew was cooked on the stove, and it tasted good

1. Encoder

Self-attention:

X . W_q = Q : query : another way to represent X : what everyone is looking for

X . W_k = K : key : yet another way to represent X : what everyone can offer

$$\underset{X}{\begin{bmatrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{bmatrix}} \cdot \underset{W\_q}{\begin{bmatrix} 1.1 & 0.6 \\ -2.8 & 2.4 \end{bmatrix}} = \underset{Q}{\begin{bmatrix} -0.995 & 3.74 \\ -3.724 & 0.6 \end{bmatrix}}$$

$$\underset{X}{\begin{bmatrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{bmatrix}} \cdot \underset{W\_k}{\begin{bmatrix} -1.7 & 0.5 \\ -1.4 & 0.9 \end{bmatrix}} = \underset{K}{\begin{bmatrix} -4.71 & 1.92 \\ 1.92 & -0.24 \end{bmatrix}}$$

Self-attention:

Compare Q and K : Q . K^T

why K^T?: if Q(4x2) and K(4x2) then Q and K should be multipliable

$$Q \cdot K^T = \boxed{\begin{matrix} 11.84 & -2.79 \\ 18.67 & -7.28 \end{matrix}}$$

$$\text{softmax}(Q \cdot K^T) = \boxed{\begin{matrix} 9.9\text{e-}1 & 1.05\text{e-}7 \\ 9.9\text{e-}1 & 2.01\text{e-}9 \end{matrix}} = \boxed{\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix}} \text{ (approx.)}$$

Self-attention:

X . W_v = V : value : yet another way to represent X : what everyone is worth

$$X \cdot W_v = V$$

$$\begin{bmatrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{bmatrix} \cdot \begin{bmatrix} 1.5 & -1.0 \\ -0.3 & -0.2 \end{bmatrix} = \begin{bmatrix} 2.478 & -2.088 \\ -2.631 & 1.606 \end{bmatrix}$$

self_attention = softmax(Q . K^T) . V

$$= \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2.478 & -2.088 \\ -2.631 & 1.606 \end{bmatrix} = \begin{bmatrix} 2.48 & -2.08 \\ 2.48 & -2.08 \end{bmatrix}$$

This could possibly suggest a one-word-translation from English to Hindi

The weights used to calculate self attention are the same for “lets” and “go”.

No matter how many words are input to the transformer, we reuse the same sets of weights for each word.

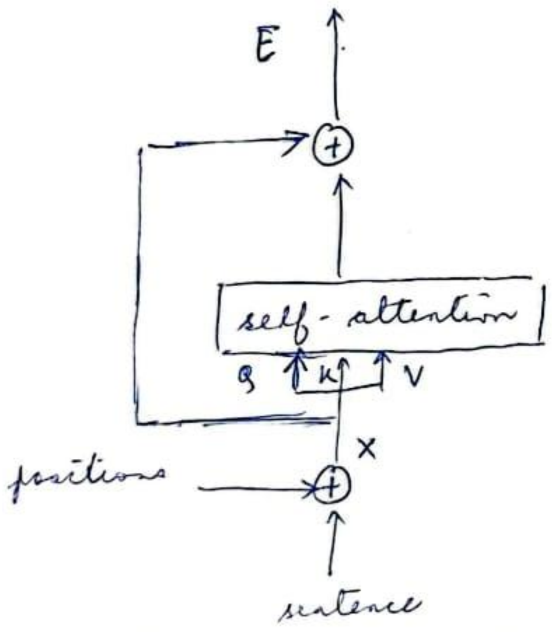
We can hence calculate all Q, K and V for all words at the same time. Was not possible with RNNs.

Residual connection:

E = self_attention + X

$$= \begin{bmatrix} 2.48 & -2.08 \\ 2.48 & -2.08 \end{bmatrix} + \begin{bmatrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{bmatrix} = \begin{bmatrix} 4.35 & -0.998 \\ 0.798 & -1.418 \end{bmatrix}$$

encoder output

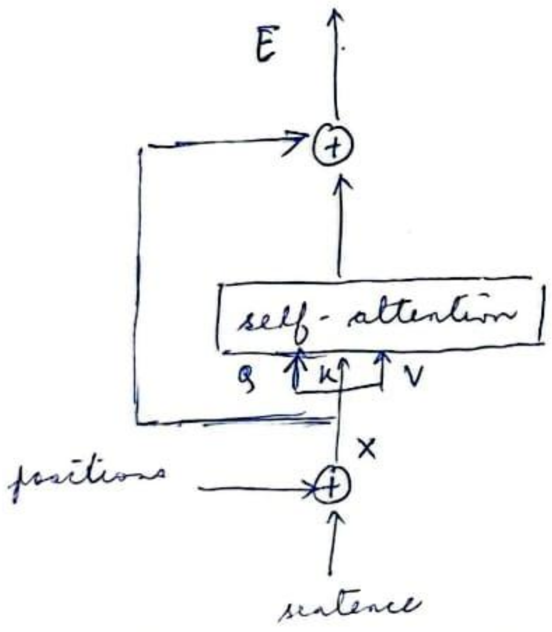


Residual connection:

$$E = \text{self\_attention} + X$$

$$= \begin{bmatrix} 2.48 & -2.08 \\ 2.48 & -2.08 \end{bmatrix} + \begin{bmatrix} 1.87 & 1.09 \\ -1.68 & 0.67 \end{bmatrix} = \begin{bmatrix} 4.35 & -0.998 \\ 0.798 & -1.418 \end{bmatrix}$$

encoder output



Residual connections help in training:
SA layer can establish a relationship with input words without having to preserve all the functions that led to it.

2. Decoder

इधर (*idhar*)	1	0	0	0
चलो (*chalo*)	0	1	0	0
हम (*hum*)	0	0	1	0
<EOS>	0	0	0	1

Hindi vocabulary = I_4 = $\boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}$

Word embedding:

$$I_4 . \boxed{\begin{matrix} -2.27 & 2.54 \\ 0.04 & 1.97 \\ -0.77 & -0.75 \\ 2.70 & -1.34 \end{matrix}} = \boxed{\begin{matrix} -2.27 & 2.54 \\ 0.04 & 1.97 \\ -0.77 & -0.75 \\ 2.70 & -1.34 \end{matrix}} \begin{matrix} \text{इधर } (\textit{idhar}) \\ \text{चलो } (\textit{chalo}) \\ \text{हम } (\textit{hum}) \\ \text{<EOS>} \end{matrix}$$

The decoder generates text iteratively, i.e., it predicts the next word based on the previous word(s). So unlike the encoder, where the entire sentence was processed together ([let's, go]), here we will process one word at a time.

We assume the first word to be <EOS> in Hindi (or any target language), since that would be the last word of the (imaginary) previous sentence, preceding the first word of the current sentence.

-2.27	2.54	इधर (*idhar*)
0.04	1.97	चलो (*chalo*)
-0.77	-0.75	हम (*hum*)
2.70	-1.34	<EOS>

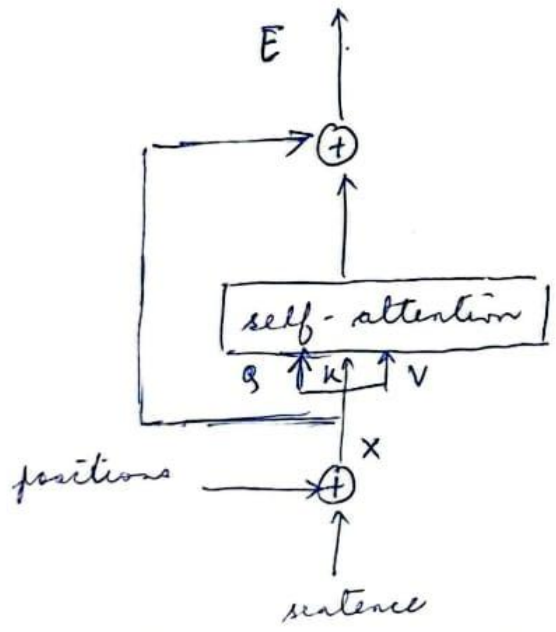
<EOS> = < 2.70, -1.34 >

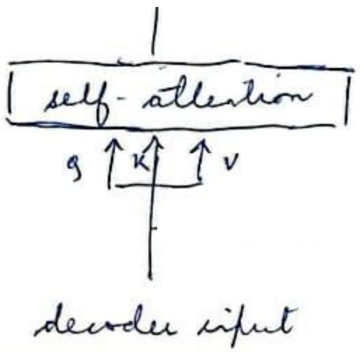
<EOS, 0> = <EOS> + <0> = < 2.70, -1.34 > + < 0, 1 >

= < 2.70, -0.34 >

recall positional embedding? we use the same embeddings here.

0	1	0
-0.9	0.4	1


E
self-attention
q
k
v
x
positions
sentence


self-attention
q
k
v
decoder input

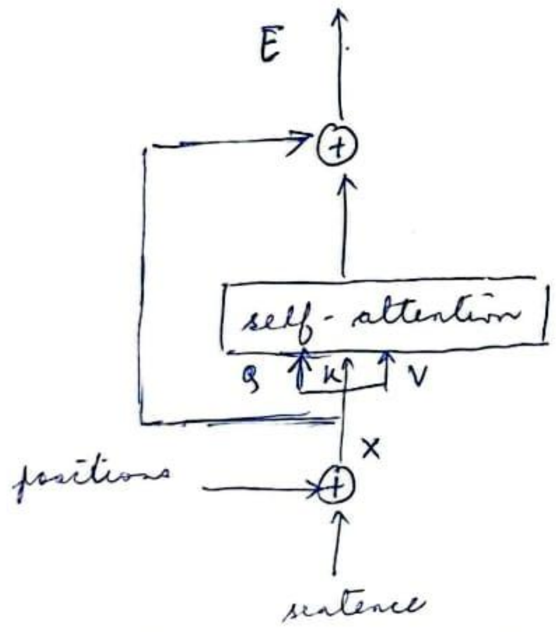
Decoder self-attention:

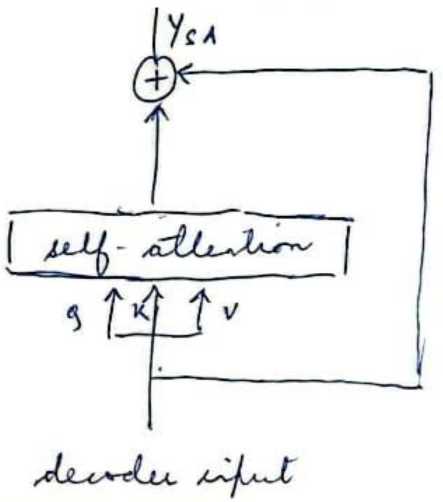
$$W\_q,sa = \begin{bmatrix} 0.4 & 0.4 \\ -0.3 & 0.1 \end{bmatrix} \qquad W\_k,sa = \begin{bmatrix} 0.4 & -0.7 \\ -0.4 & -0.3 \end{bmatrix} \qquad W\_v,sa = \begin{bmatrix} -1.1 & -0.7 \\ -0.4 & 1.3 \end{bmatrix}$$

<EOS, 0> . W_q,sa = Q_sa
<EOS, 0> . W_k,sa = K_sa
<EOS, 0> . W_v,sa = V_sa

decoder_self_attention = softmax(Q_sa . K_sa^T) . V_sa

= < -2.834, -2.332 >


E
self-attention
q
k
v
x
positions
sentence

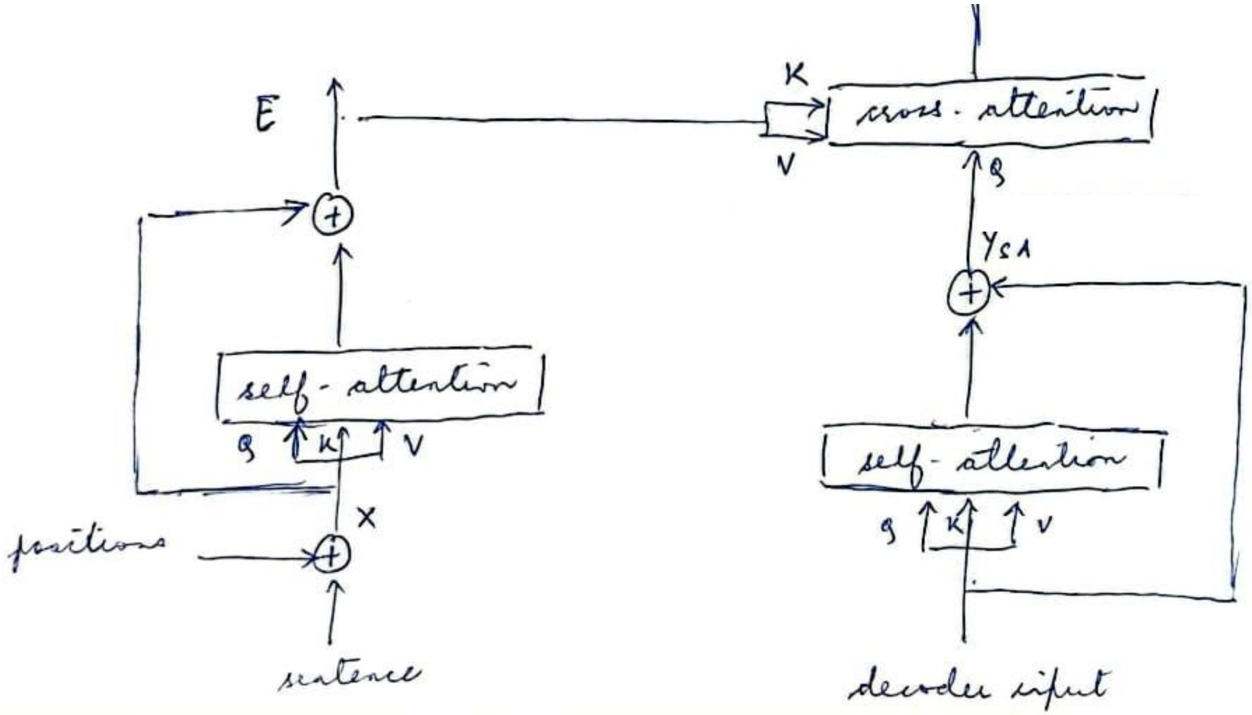

self-attention
q
k
v
decoder input

Residual network:

Y_sa = decoder_self_attention + <EOS, 0>

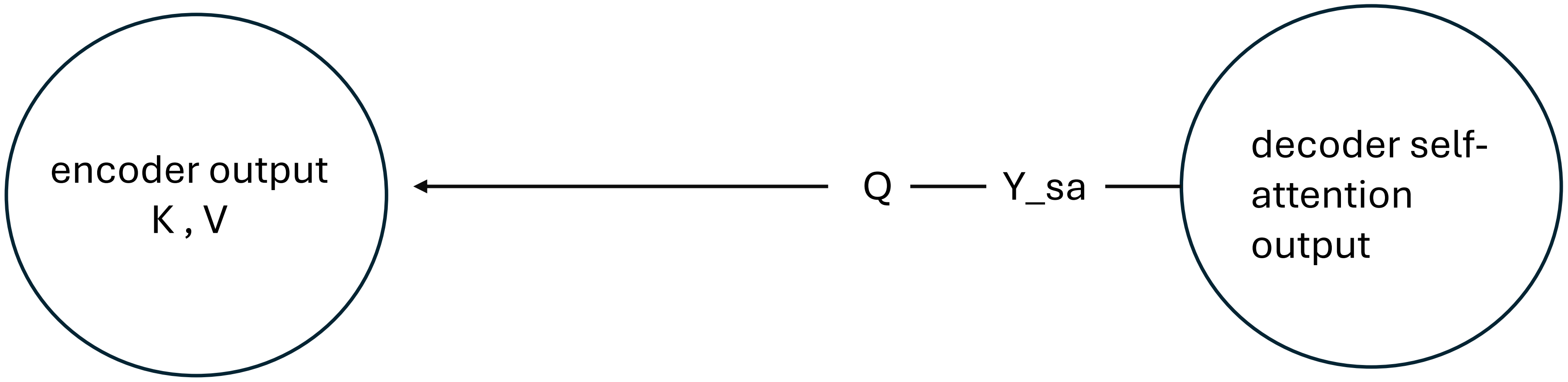
= < -2.834, -2.332 > + < 2.70, -0.34 >

= < -0.134, -2.672 >


E
K
cross-attention
V
Q
Y_{S1}
self-attention
Q
K
V
X
positions
sentence
self-attention
Q
K
V
decoder input

Decoder cross-attention:

We have now learnt the meaning of the current word w.r.t the other words in Hindi (through self attention)
But what does the current word mean w.r.t the English that was learnt by the encoder?



Decoder cross-attention:

$$W\_q,ca = \begin{bmatrix} 1.5 & -0.3 \\ 0.3 & -1.0 \end{bmatrix} \qquad W\_k,ca = \begin{bmatrix} -1.1 & 0.3 \\ -0.3 & -0.8 \end{bmatrix} \qquad W\_v,ca = \begin{bmatrix} 1.1 & 0.6 \\ -1.2 & -0.5 \end{bmatrix}$$

Query from decoder self attention output:

Y_sa . W_q,ca = Q_ca = < -1.0026, 2.7122 >

Recall E, the encoder output?

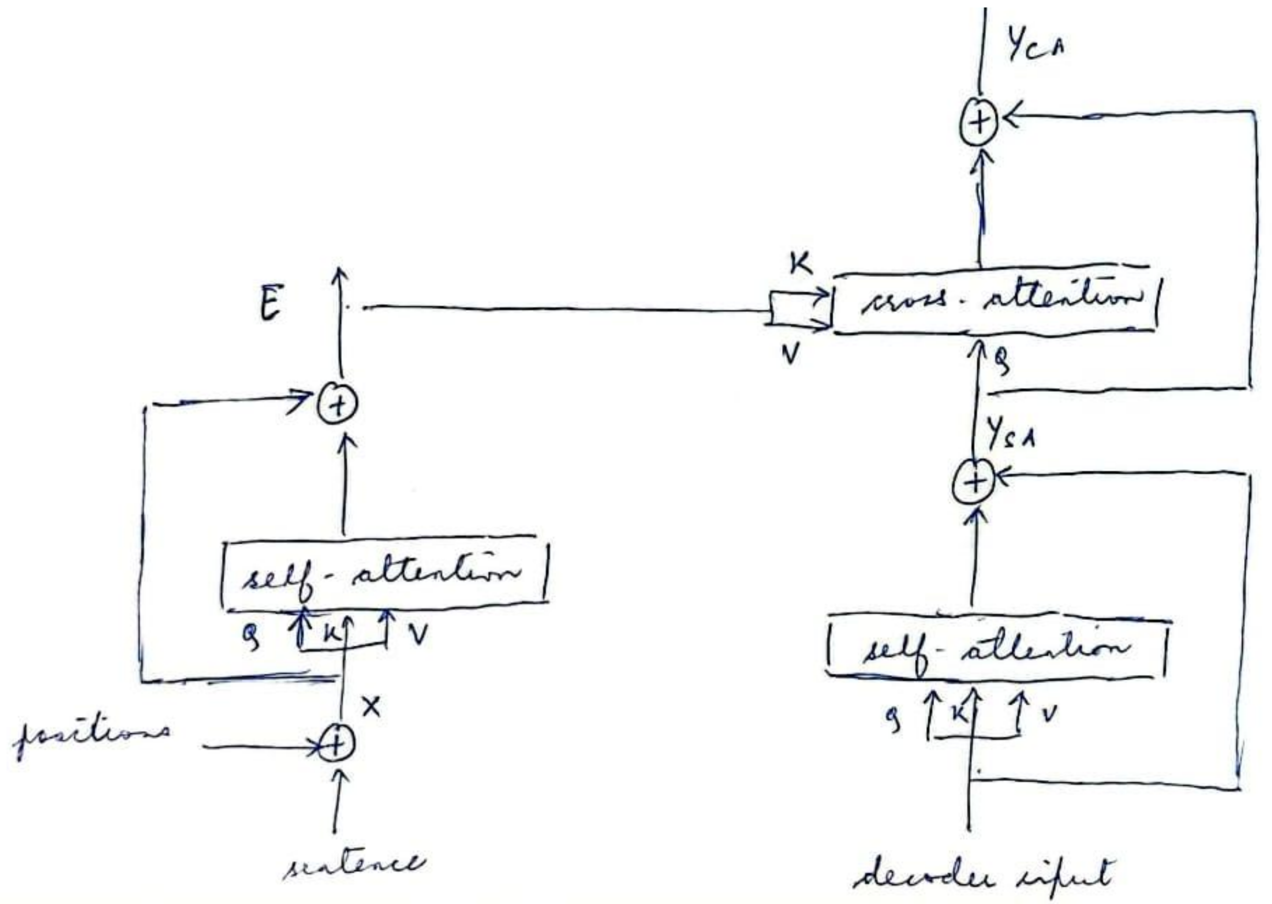
$$\begin{bmatrix} 4.35 & -0.998 \\ 0.798 & -1.418 \end{bmatrix}$$

Key and Value from encoder output:

$$E\,.\,W\_k,ca = K\_ca = \begin{bmatrix} -4.48 & 2.10 \\ -0.45 & 1.37 \end{bmatrix}$$

$$E\,.\,W\_v,ca = V\_ca = \begin{bmatrix} 5.98 & 3.11 \\ 2.58 & 1.19 \end{bmatrix}$$

decoder_cross_attention = softmax(Q_ca . K_ca^T). V_ca = < 5.97, 3.10 >



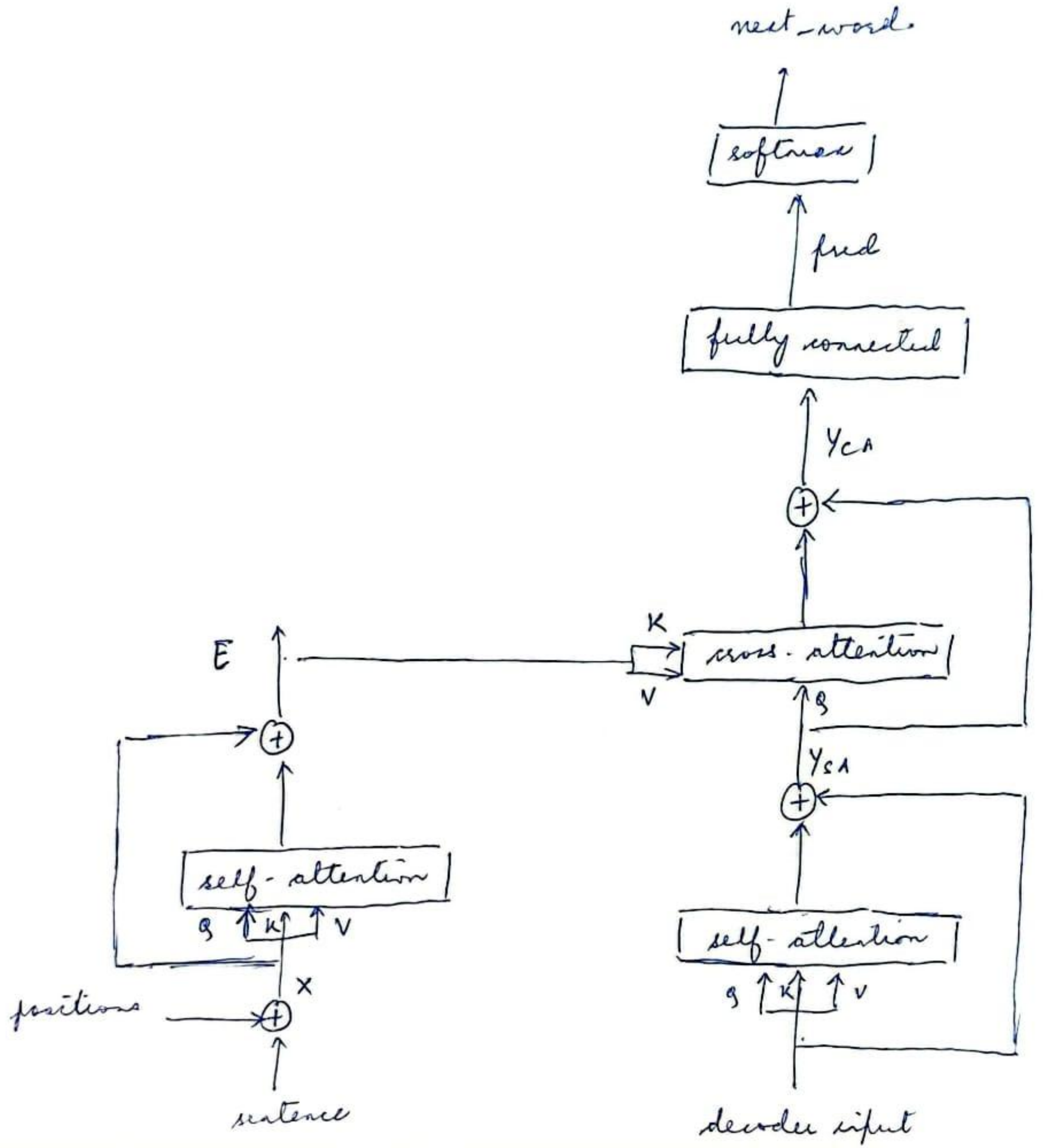
YCA
K
cross - attention
V
Q
E
YSA
self - attention
Q
K
V
X
positions
self - attention
Q
K
V
sentence
decoder input

Residual network:

Y_ca = decoder_cross_attention + Y_sa

= < 5.97, 3.10 > + < -0.134, -2.672 >

= < 5.84, 0.43 >



next_word
softmax
pred
fully connected
Y_{CA}
K
E
cross - attention
V
Q
Y_{SA}
self - attention
Q
K
V
self - attention
Q
K
V
X
positions
sentence
decoder input

Fully-connected layer

$$W = \boxed{\begin{matrix} -0.6 & 0.8 & -0.1 & -1.0 \\ -2.0 & -0.9 & -1.1 & 1.6 \end{matrix}}$$

bias = < -0.6, 1.4, -2.5, 0.5 >

pred = Y_ca . W + bias

= < -4.97 , 5.68 , -3.56 , -4.65 >

next_word = softmax(pred)

= < 2.4e-5 , 9.99e-1, 9.7e-5, 3.3e-5 > = < 0, 1, 0, 0 > (approx.)

चलो (*chalo*)

Now, चलो (*chalo*) enters the decoder, and the same decoder operations are repeated.

```
[46] chalo = hindi_words_embedded[1]
     chalo
```

```
array([0.04, 1.97])
```

```
[47] chalo = chalo + W_pos_embedding[1] # 1st index
     chalo = chalo.reshape(1,2)
     chalo
```

```
array([[-0.86,  2.37]])
```

```
[48] Q_chalo = chalo @ W_q_dsa
     K_chalo = chalo @ W_k_dsa
     V_chalo = chalo @ W_v_dsa
```

```
[49] e = Q_chalo @ K_chalo.T
     softmax_output = np.exp(e) / np.sum(np.exp(e))
     softmax_output = np.array(softmax_output)
     self_attention_chalo = softmax_output @ V_chalo
     self_attention_chalo
```

```
array([[-2.000e-03,  3.683e+00]])
```

```
[50] chalo = self_attention_chalo + chalo
     chalo
```

```
array([[-0.862,  6.053]])
```

```
[51] Q_chalo = chalo @ W_q_dca
     dot_product_ca = Q_chalo @ K_encoder_output.T
     similarity = softmax(dot_product_ca)
```

```
[52] cross_attention = similarity @ V_encoder_output
```

```
[53] chalo = cross_attention + chalo
```

```
[54] pred = chalo @ W_fcl + bias
```

```
[55] np.argmax(softmax(pred))
```

```
3
```

3 = EOS

Next word predicted is <EOS>
Translation finished.