Summary of Transformers

* Attention

Lets say Q is a query vector
K_i is a key vector
V_i is a value vector

Every key vector is associated with a value vector

Types

(1) Additive Attention

$$s_i = w_3 \tanh\left(w_2^T Q + w_1^T K_i \right)$$

This equation can be related to the attention seen in RNNs

$$f_{ATT} = V^T \tanh\left(\underbrace{U S_{t-1}}_{\text{decoder state}} + \underbrace{W h_j}_{\text{encoder state}} + b \right)$$

(2) Dot-Product Attention

$$s_i = Q^T K_i$$

Here in the above types s_i is the similarity between Q & K_i

Now,

$$A(Q, K, V) = \sum_i \frac{e^{s_i}}{\sum_j e^{s_j}} V_i$$

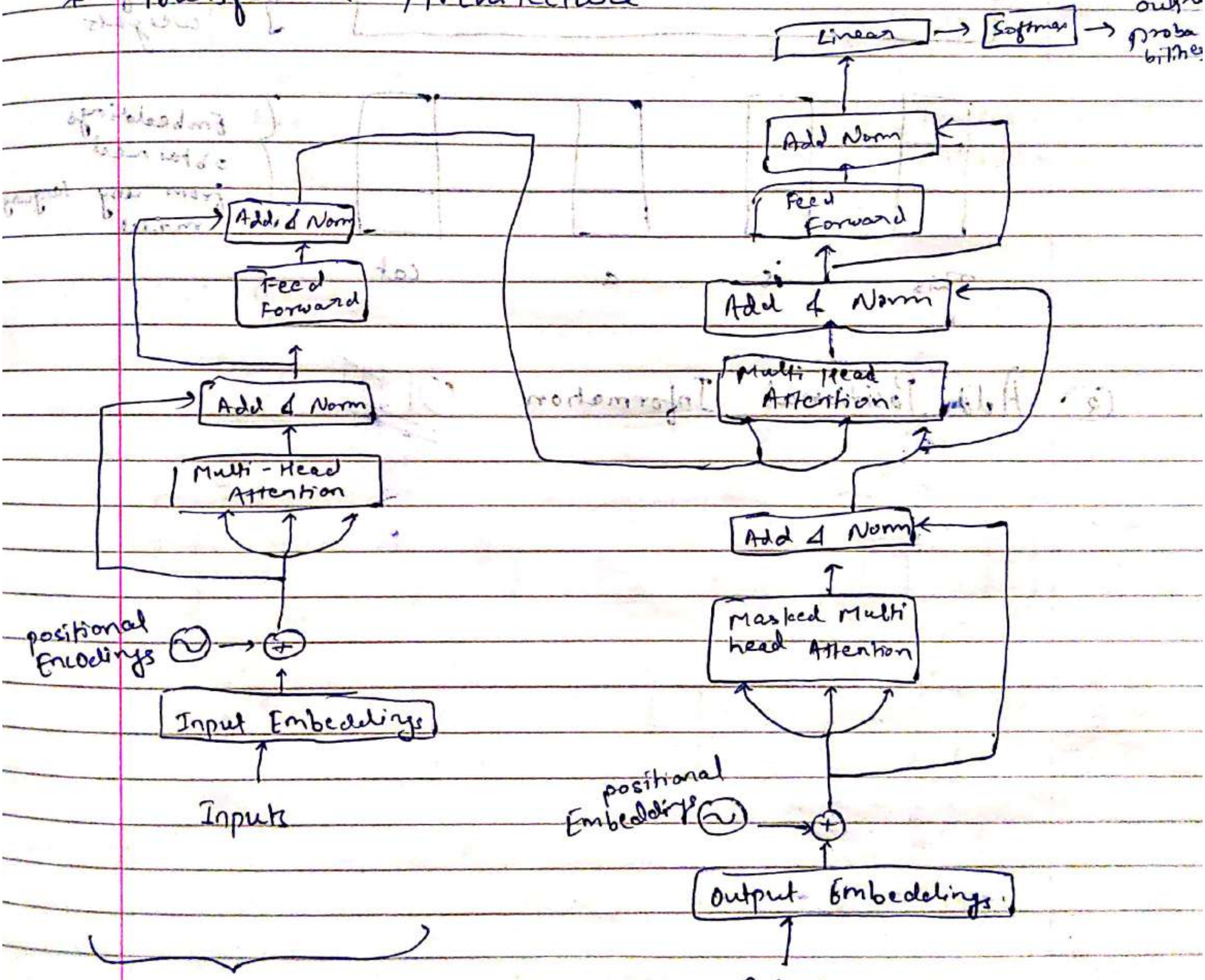
PAGE NO.:
DATE:

$= \underbrace{\text{softmax}(s_i)}_{} \cdot V_i$

softmax of s_i (Q & k_i)

over (s_1 (Q & k_1), s_2 (Q & k_2) .....)

* Transformer Architecture

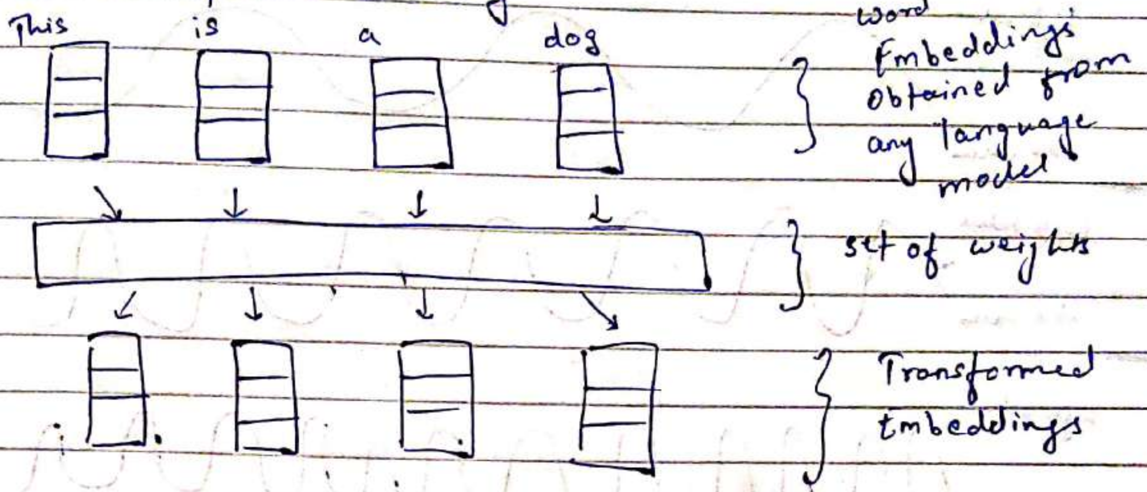


Encoder

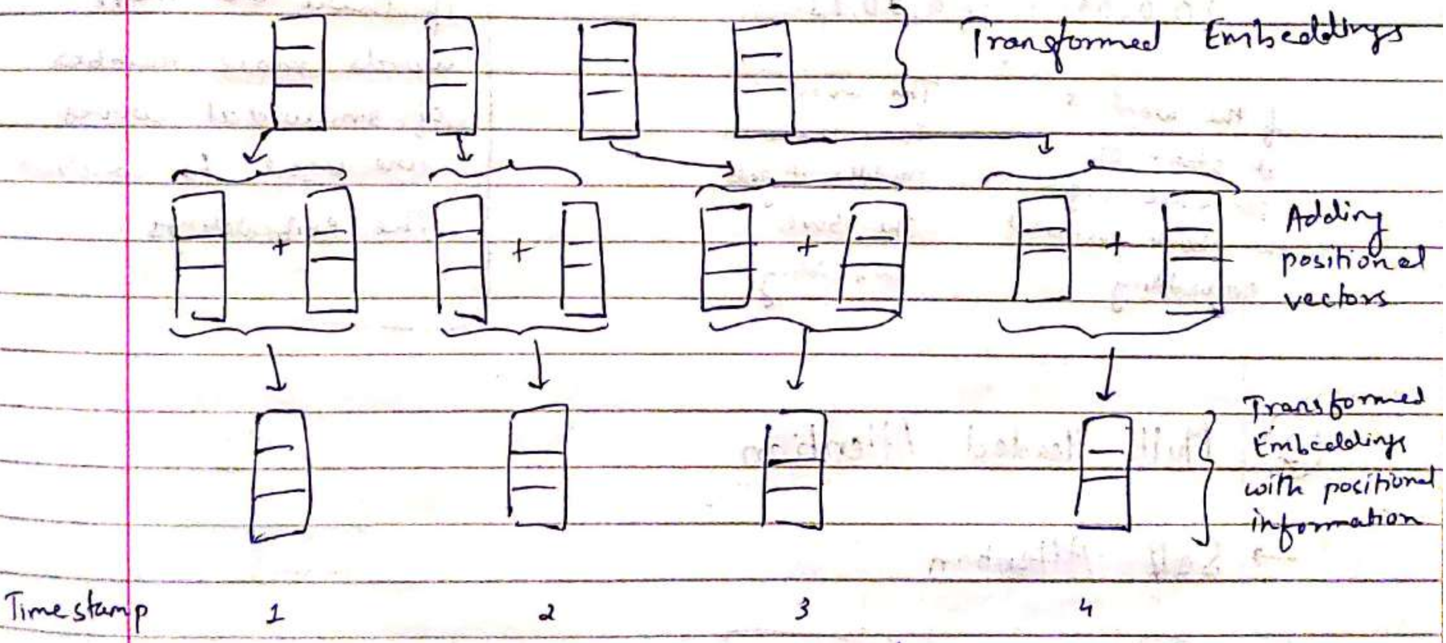
Decoder

→ Encoder

① Construct input Embedding



② Add Positional Information

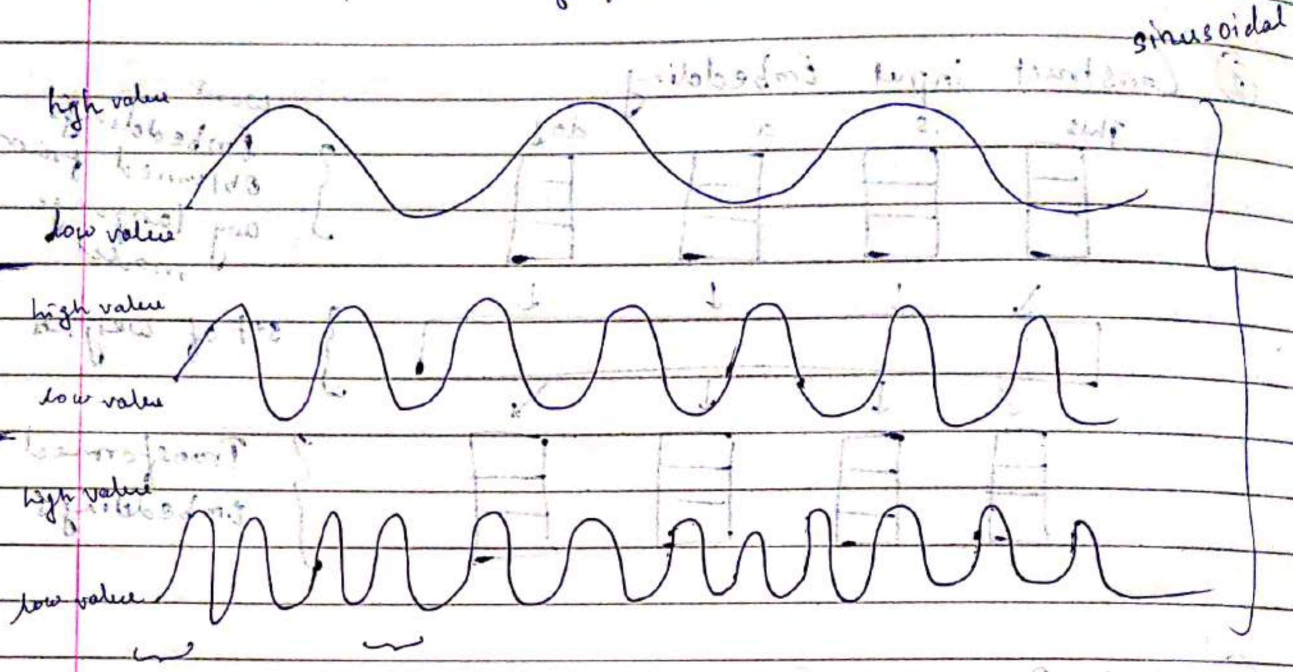


$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dmodel}}\right)$$ Odd timestamp

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$ Even timestamp

PAGE NO.
DATE:

→ Another interpretation of positional vectors



$[0,0,0]$

If the word is at start of sentence it gets the above positional embedding

$[0,0,1]$

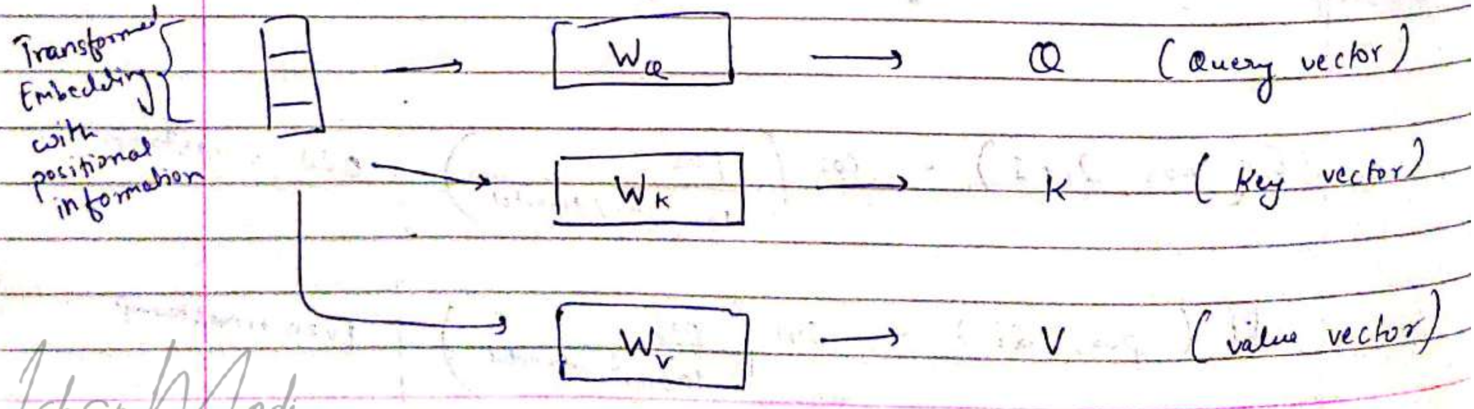
The word is somewhere in middle it gets the above embedding

Note

If there are more words more number of sinusoidal waves are used to construct the embeddings

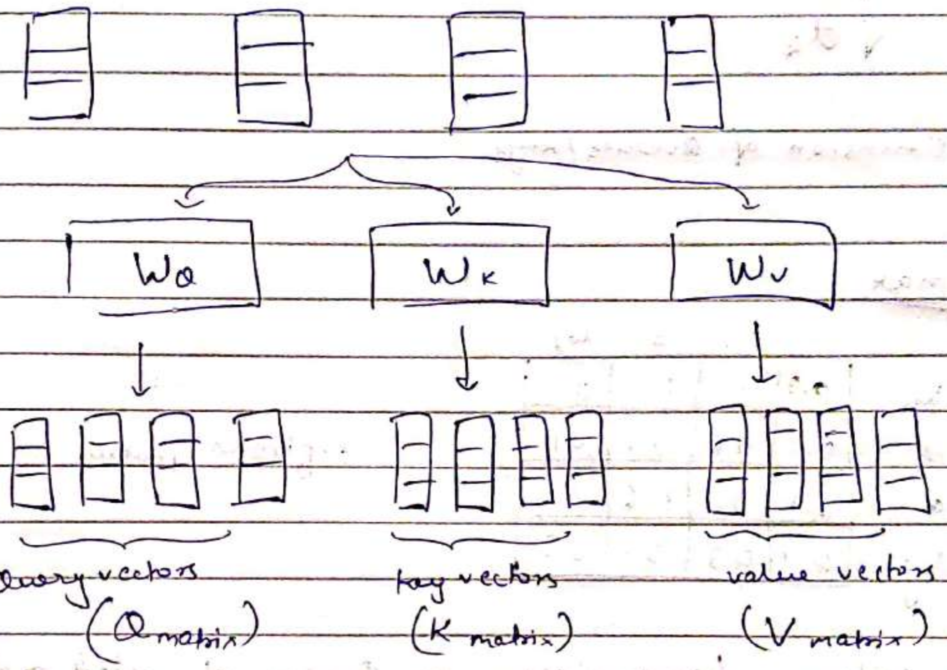
(3) Multi Headed Attention

→ Self Attention



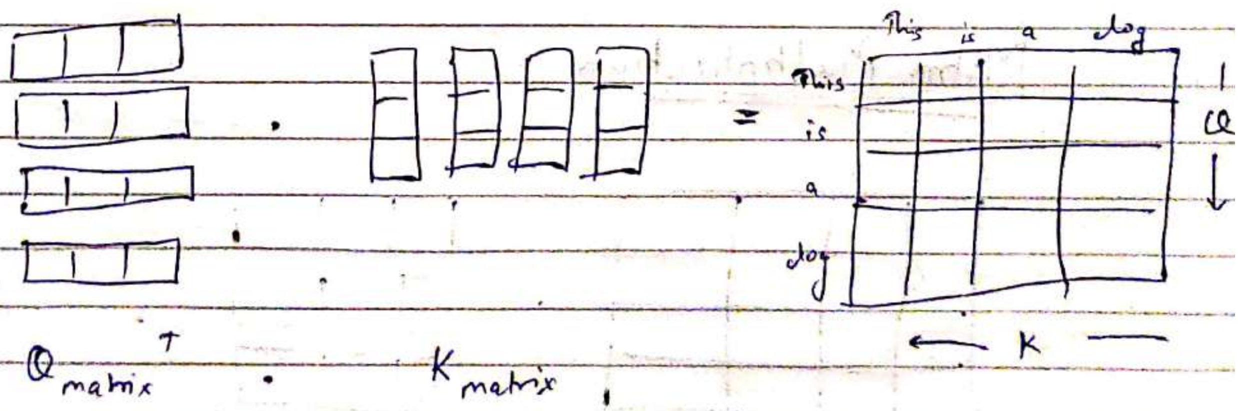
This is called self attention because Q, K & V are generated from input embeddings

For multiple vectors



→ Matrix Multiplication

Every ~~key vector~~ Query vector is multiplied by all key vectors



PAGE NO.:
DATE:

The matrix obtained after multiplication tells us what is importance of key (column) in the query (row)

Scale

$$\frac{\text{Obtained matrix}}{\sqrt{d_k}} = \text{scaled Matrix}$$

d_k → dimension of Queries / keys

Softmax

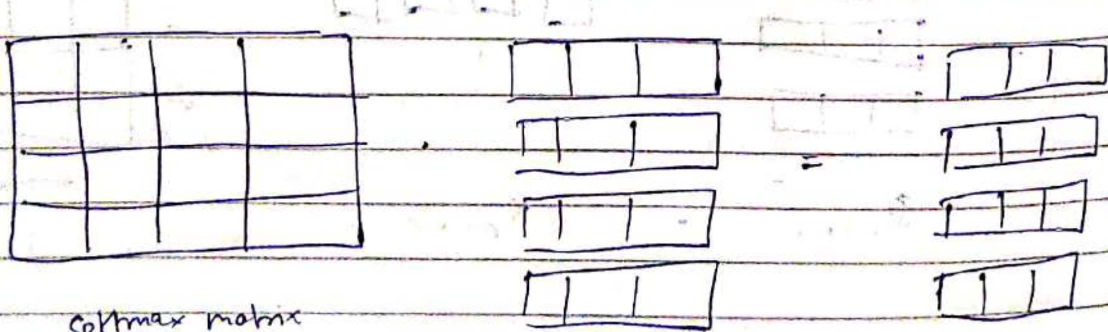
	This	is	a	dog
This	0.7	0.1	0.1	0.1
is	0.1	0.6	0.2	0.1
a	0.1	0.3	0.6	-0.1
dog	0.1	0.3	0.3	0.3

softmax matrix

Apply softmax horizontally on Query of scaled matrix to get above matrix.

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Matrix Multiplication



softmax matrix · V_{matrix}^T = $Att\ matrix^T$

Date: / /

Till now,

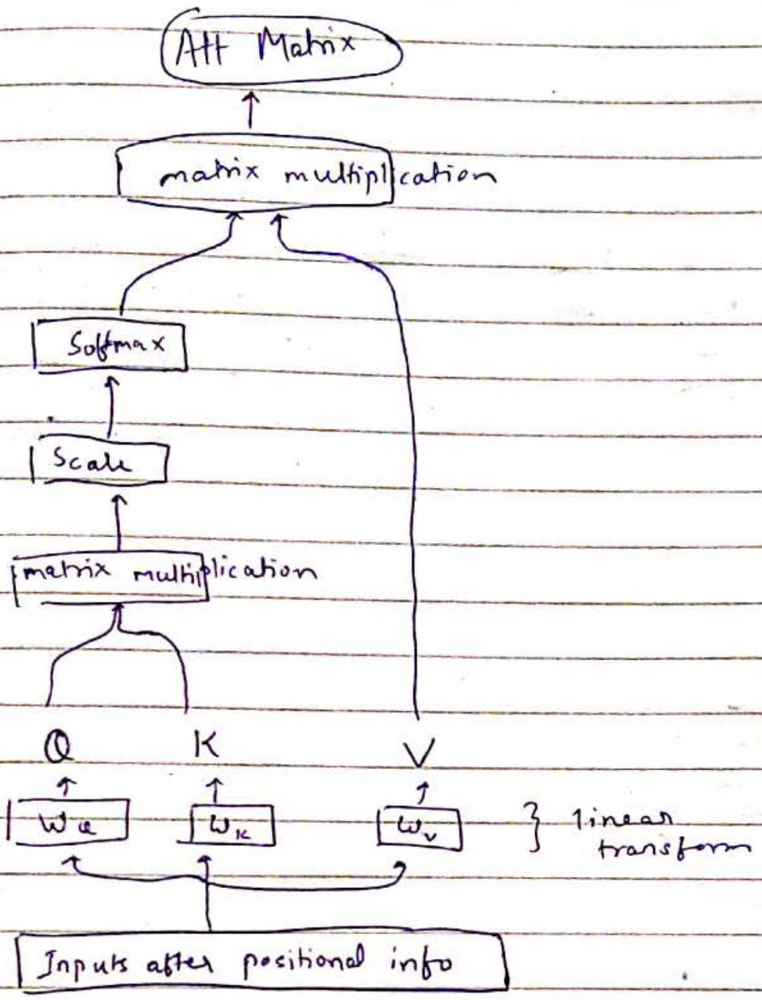
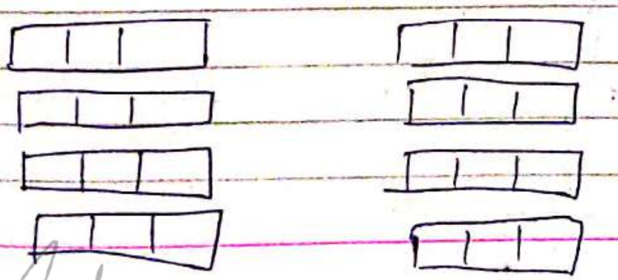


Figure 1

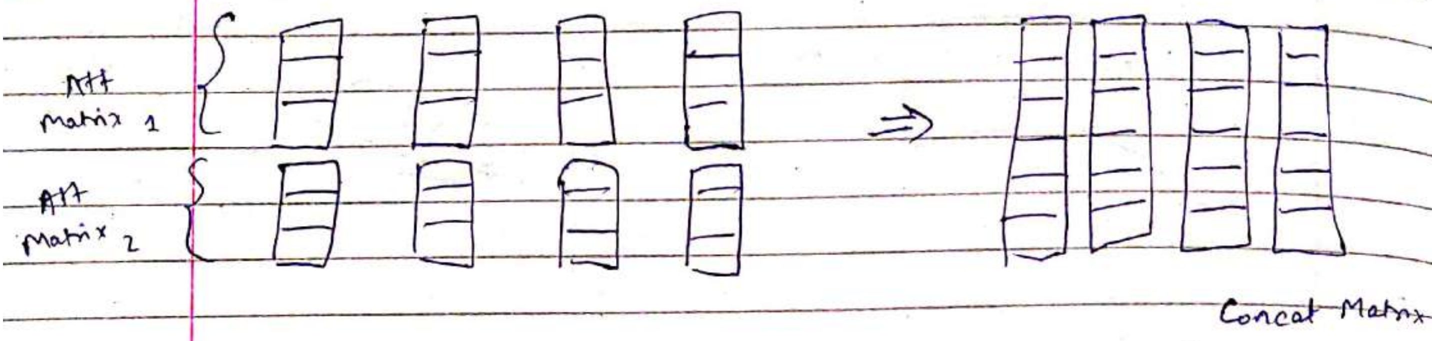
→ Above is one attention module. lets say we have multiple W_Q, W_K, W_V and thus multiple attention modules

In this case we would have multiple Att Matrix

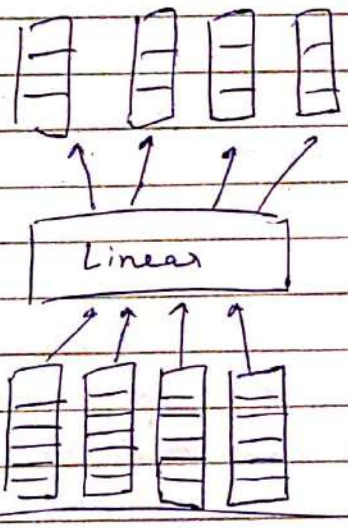
Att Matrix_1^T Att Matrix_2^T



Concatenation

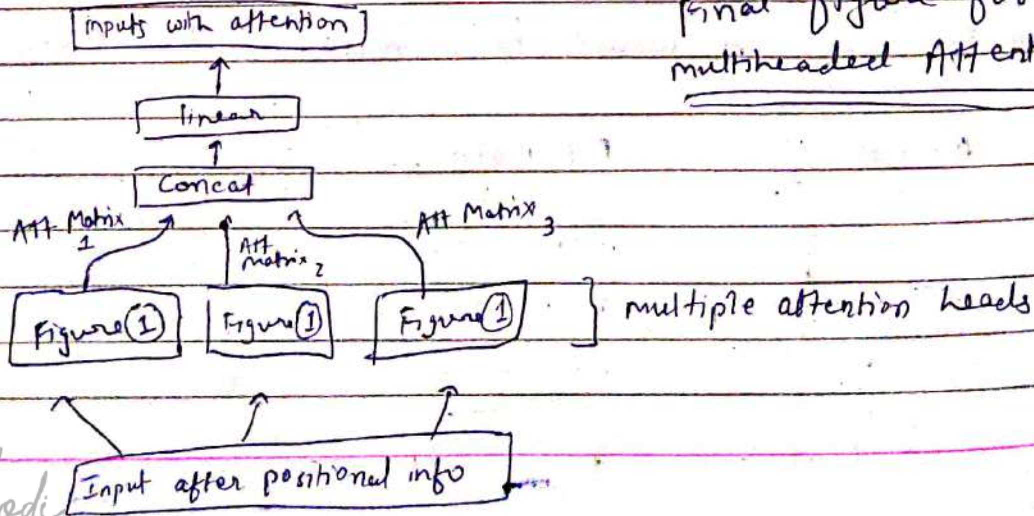


Linear



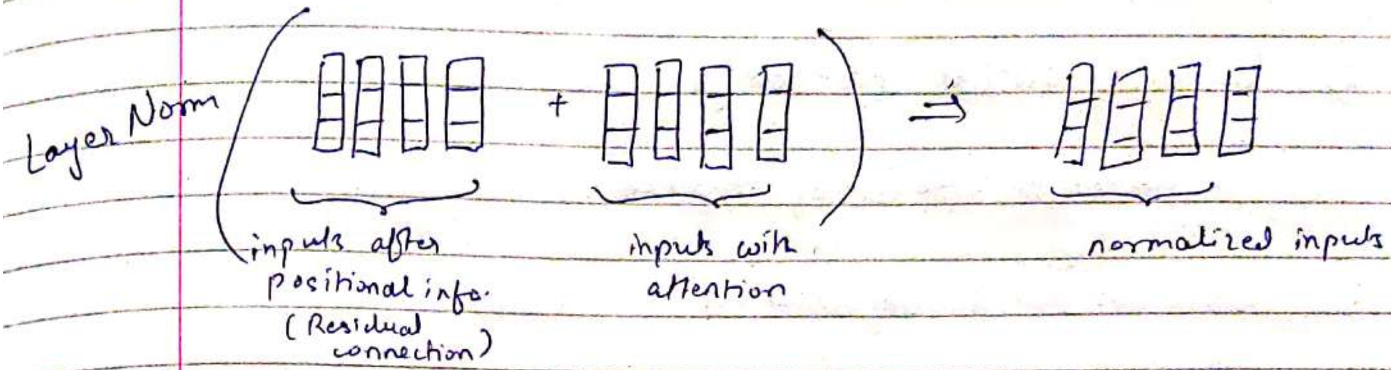
Thus

Final figure for multiheaded Attention



Date: / /

(4) Add & Normalize



(5) Feed Forward

normalized inputs

↓

Linear

↓

ReLu

↓

Linear

↓

Feed Forward Component

Here We have ~~discussed~~ discussed all the components used in encoder of transformers.

→ Decoder

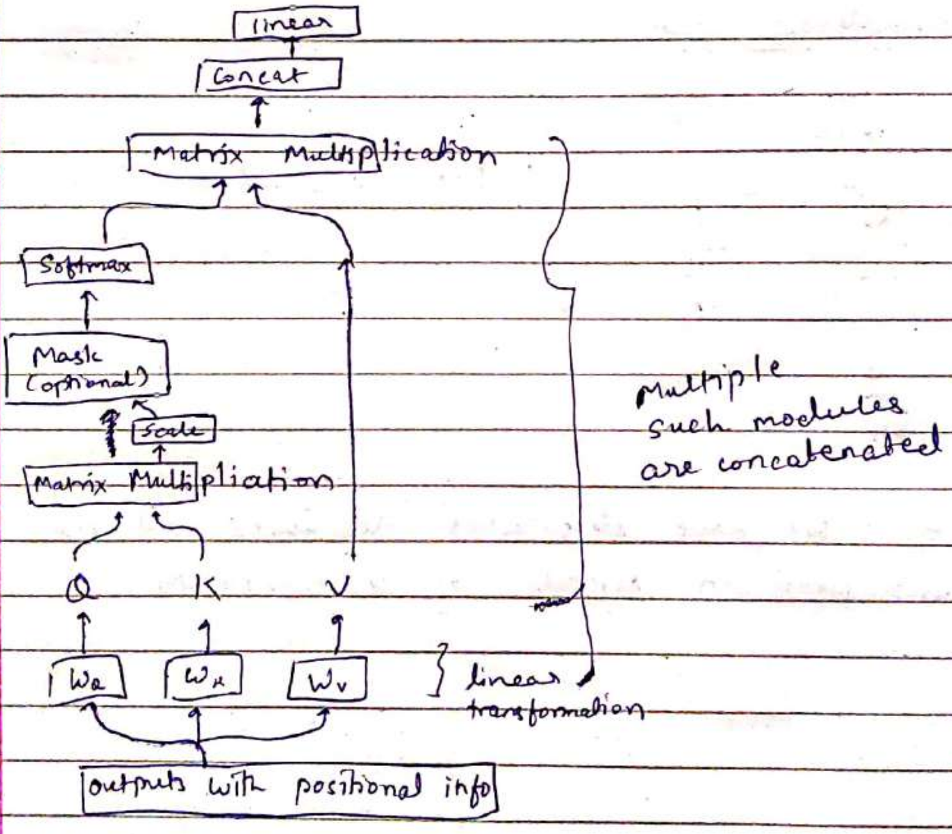
1. Construct output Embeddings

similar to step (1) of encoder

2. Add Positional Information

similar to step (2) of encoder

3. Masked Multi Headed Attention



The only difference here is the Mask from the Attention structure in encoder

Mask

Lets say we have the following score matrix after scoring

0.7	0.1	0.1	0.1
0.1	0.6	0.2	0.1
0.1	0.3	0.6	0.1
0.1	0.3	0.3	0.3

+

0	$-\infty$	$-\infty$	$-\infty$
0	0	$-\infty$	$-\infty$
0	0	0	$-\infty$
0	0	0	0

mask values of future

=

Q ↓ / K →	This	is	a	dog
This	0.7	$-\infty$	$-\infty$	$-\infty$
is	0.1	0.6	$-\infty$	$-\infty$
a	0.1	0.3	0.6	$-\infty$
dog	0.1	0.3	0.3	0.3

Since transformer is auto regressive (it predicts future based on past) we need to hide the future information so that it can learn & predict correctly.

Here in this case,

softmax (Masked Matrix) =

1	0	0	0
0.37	0.62	0	0
0.26	0.31	0.43	0
0.21	0.26	0.26	0.26

Thus attention is not given to future because we want to predict the future.

Date: / /

(4) Multi Headed Attention

→ **Cross Attention**

Works similar to self attention.

It is called cross attention because

v. imp
- key and values come from encoder
- query come from masked multi head attention of decoder

(5) Linear Classifier

Decoder is followed by linear classifier & softmax to give a probability distribution

probability distribution (N class) vocab size

↑

$\boxed{\text{Softmax}}$

↑

vector (N class) vocab size

↑

$\boxed{\text{Linear Classifier}}$

↑

Note -