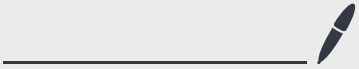
Attention is all you need

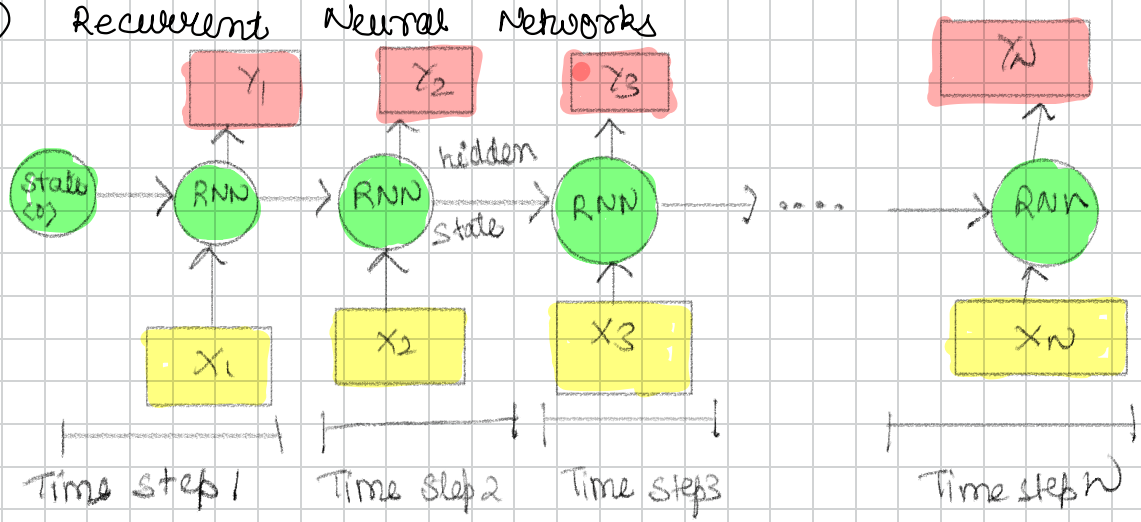


Attention is all you need

GPT: Generatively Pre-Trained Transformers

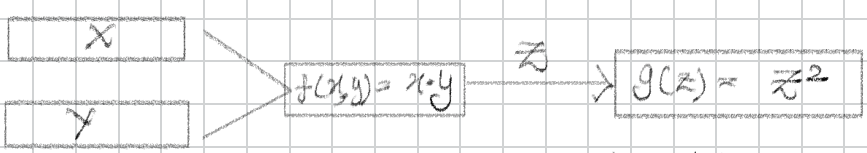
Recurrent Neural Network ⟶ Transformers

i) Recurrent Neural Networks



one series of input to get one series of output

i) Slow computation for long sequences
ii) Vanishing or exploding gradients

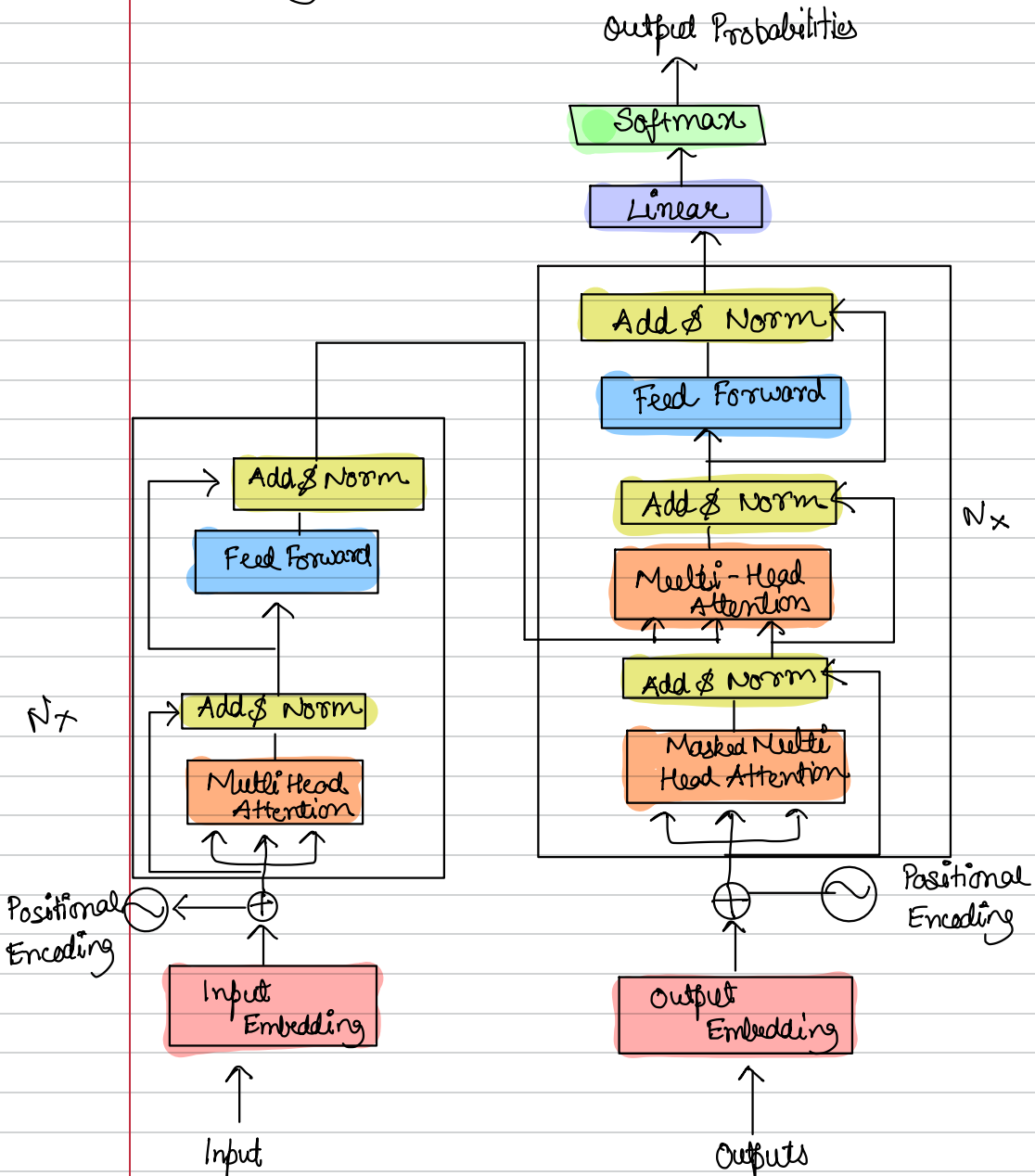


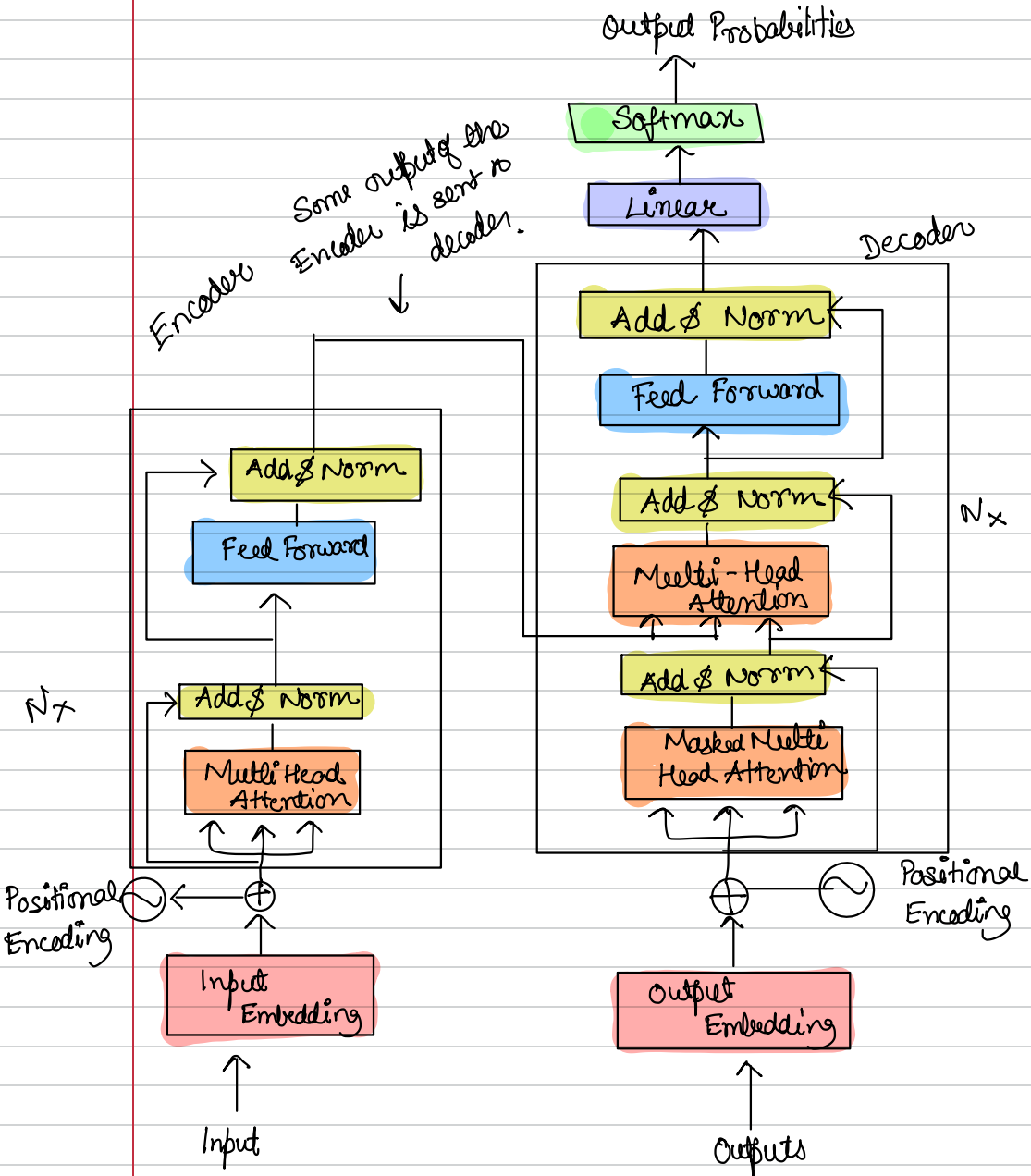
$$\frac{dg}{dx} = \frac{dg}{df} \cdot \frac{df}{dx}$$

longer this multiplication chain
longer computation time.
it'll either be a very large number or a very small number.
exploding gradient
vanishing gradient

iii) Difficulty in accessing information from long term ago.

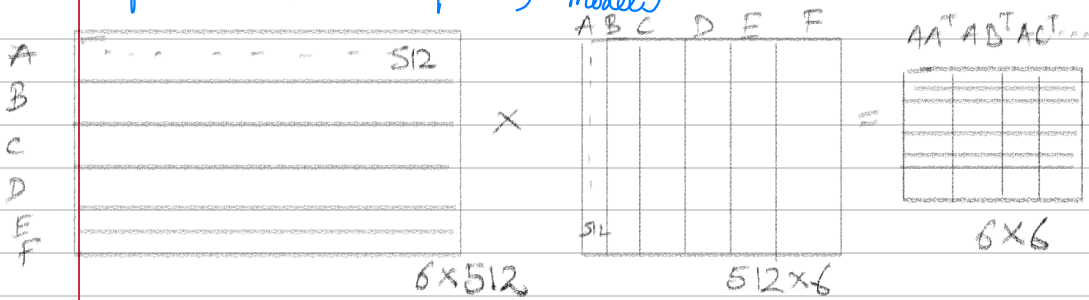
Introducing the Transformer



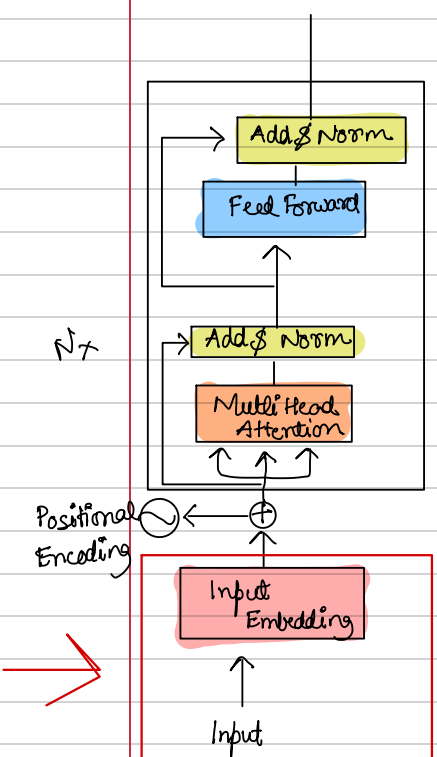

Output Probabilities
Softmax
Linear
Some output of the Encoder is sent to decoder.
Encoder
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Input
Outputs

Notations

Input Matrix (sequence, d_{model})



ENCODER



What is an Input Embedding?

original sentence (tokens)

YOUR CAT IS A LOVELY CAT

Input IDs (position in Vocab)

105 6587 5475 3578 65 6587

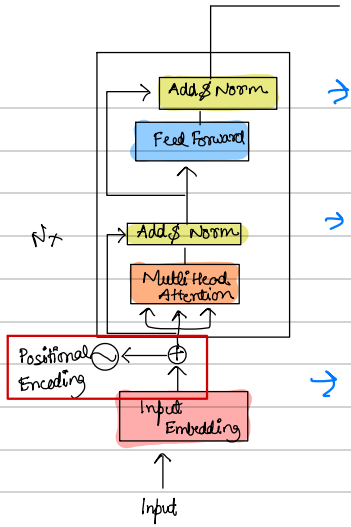
Embedding (vectors of size 512,)

Always map the same word to same embedding.

LEARN THOSE PARAMETERS.

$d_{model} = 512$

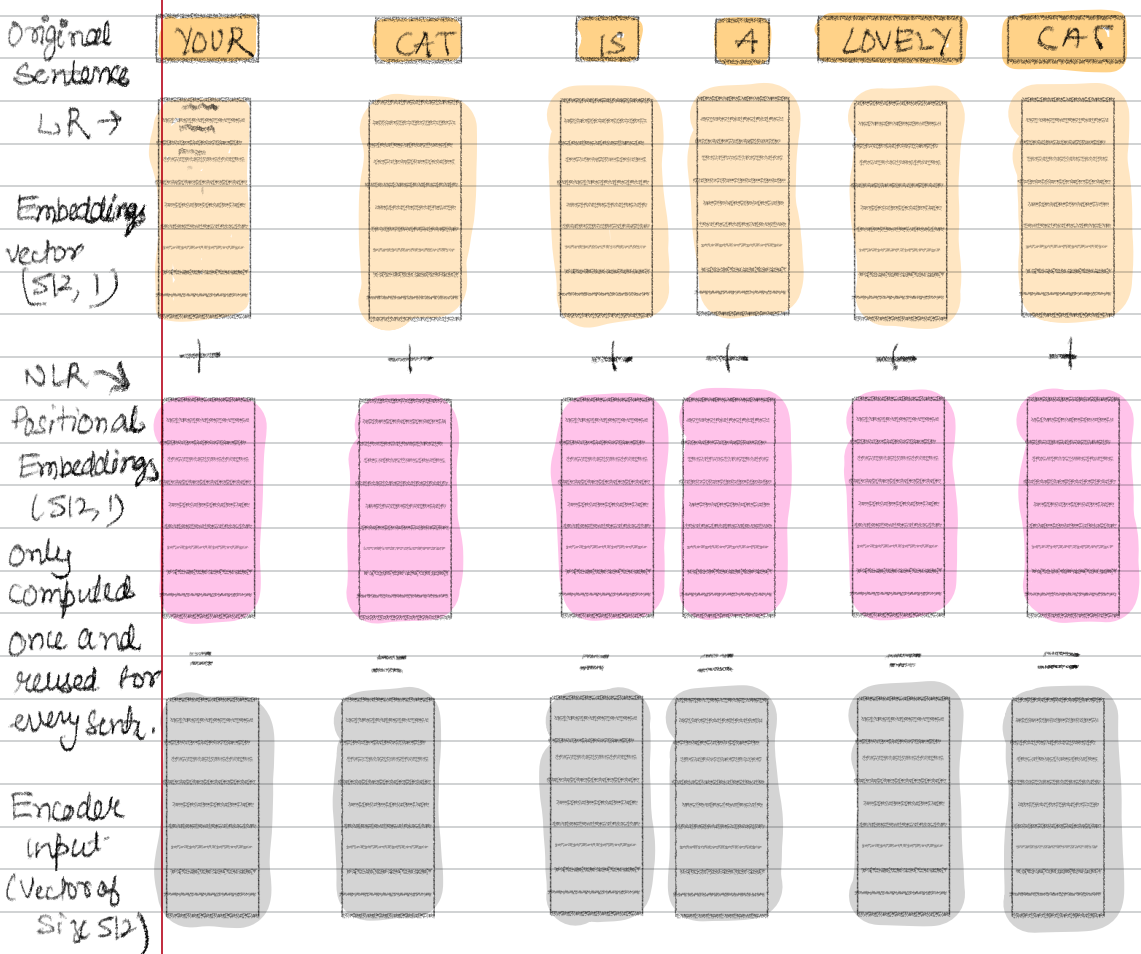
What is Positional Encoding?



→ We want each word to carry some information about its position in the sentence.

→ We want models to treat words that appear close to each other as "close" and words that are distant as "distant".

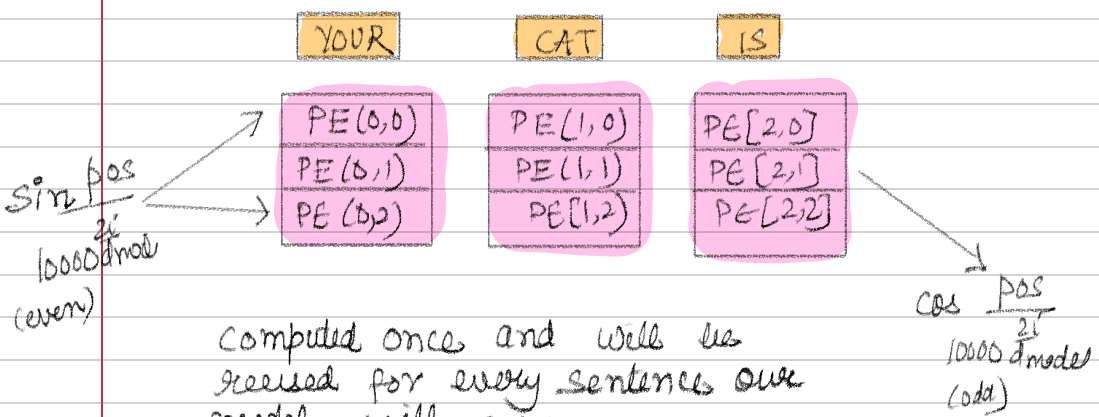
→ We want the positional encoding to represent a pattern that can be learned by the model.



How are Positional Embeddings calculated?

Embedding vector is something that we learn, while the positional Embedding remain fixed.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \qquad PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

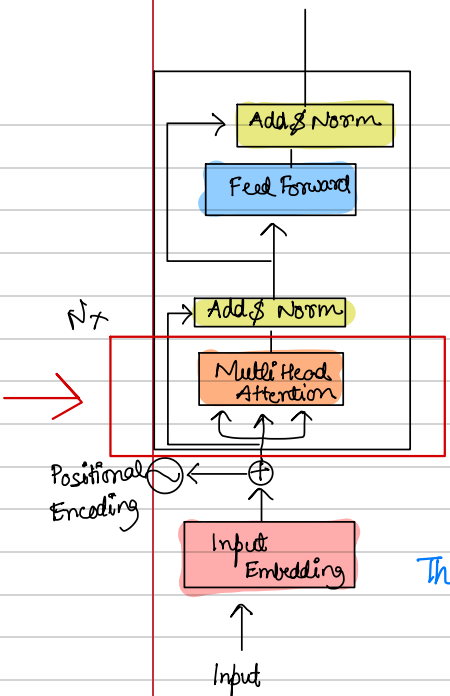


computed once and will be reused for every sentences our model will see!

We only need to compute positional encoding once and then reuse them for every sentence, no matter if it is training or inference.

Why trigonometric functions?

Trigonometric functions like cosine and sine naturally represent a pattern that the model can recognize as continuous, so relative positions are easier to see for the model.



Multi-Head Attention

Let's first understand the single head version.

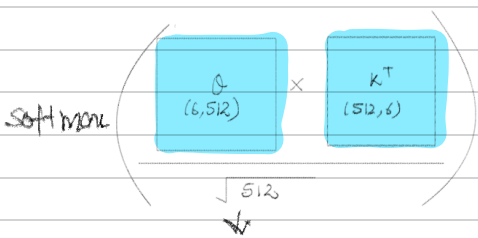
What is self Attention?

Self Attention allows the model to relate words to each other.

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

Let's assume we have $seq = 6$, $d_{model} = 512$

The matrices Q, K and V are just the input sentences.



	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.271	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.163	0.227	0.134	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

6×6

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.139	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.286	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.163	0.227	0.134
CAT	0.195	0.114	0.203	0.103	0.157	0.229

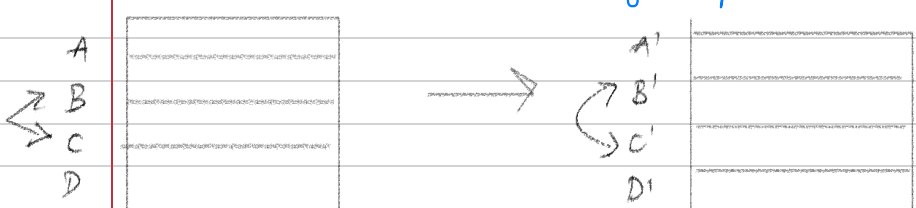
6×6

× V (6, 512) = Attention (6, 512)

Each row in the attention matrix captures not only the meaning (given by embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other word.

Self Attention in detail

- Self Attention is permutation invariant, if we don't consider the contribution of the positional encodings.



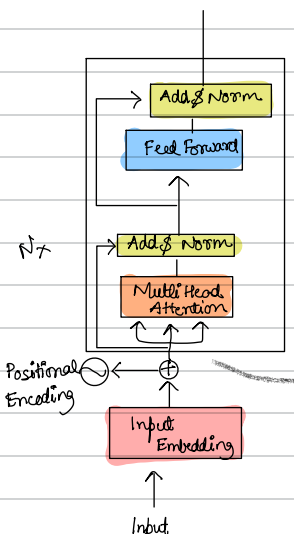
- Self Attention requires no parameters. Up to now the interaction between words has been driven by their embedding and the positional encoding. Will change later.
- We expect values along the diagonal to be the highest.
- If we don't want some positions to interact, we can always set their values to $-\infty$ before applying the softmax in this matrix and the model will not learn those interaction

Multi - Head Attention

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

$$\text{Multi Head}(Q,K,V) = \text{Concat}(\text{head}_1, \dots \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



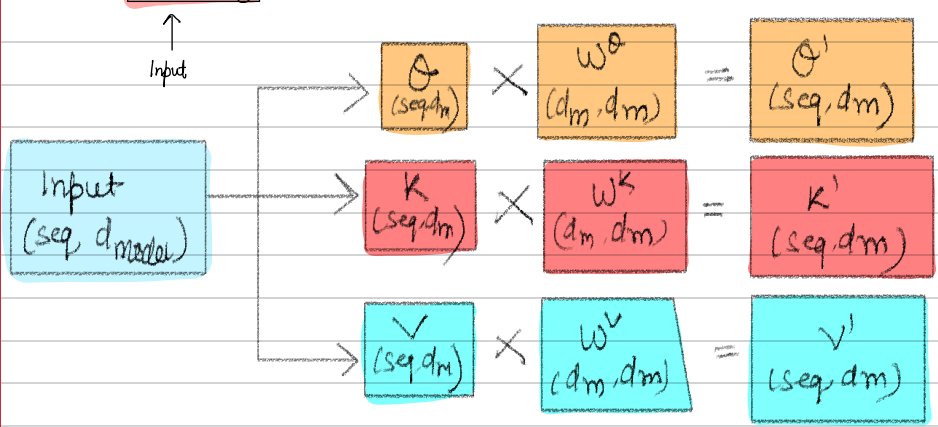
seq = sequence length

d_{model} = size of embedding vector

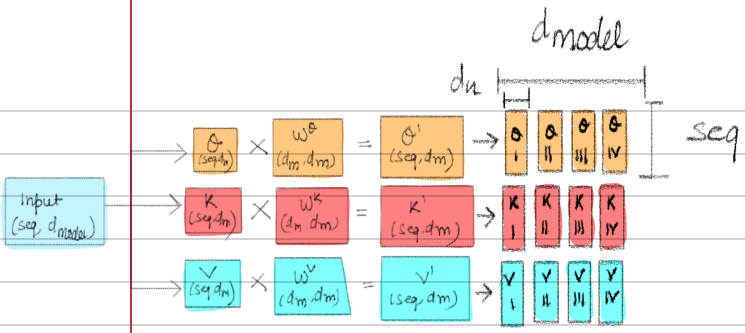
h = number of heads

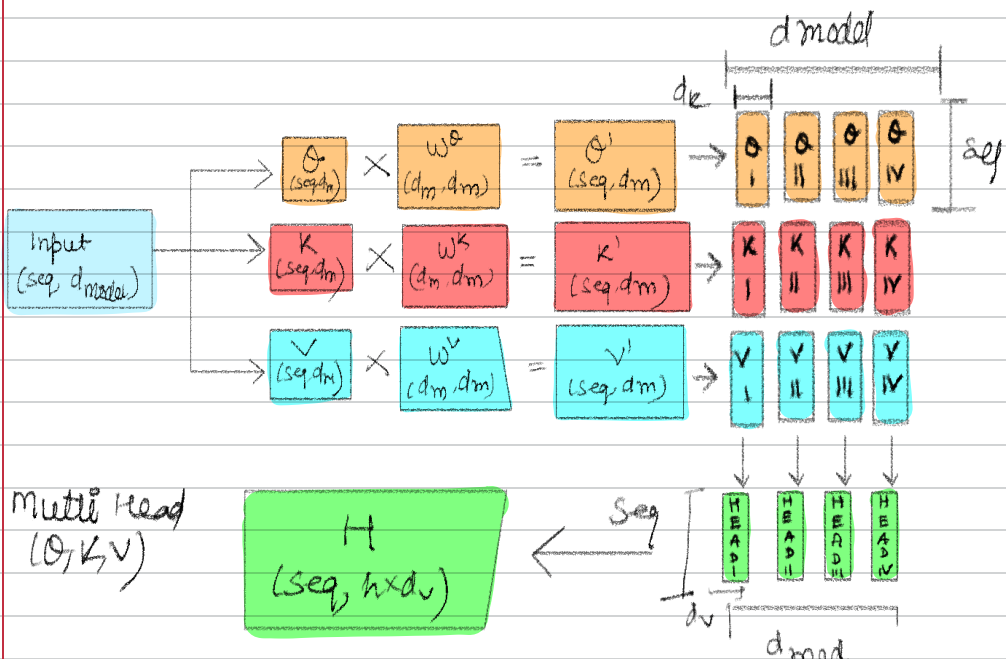
$d_k = d_v = d_{model}/h$

making four copies



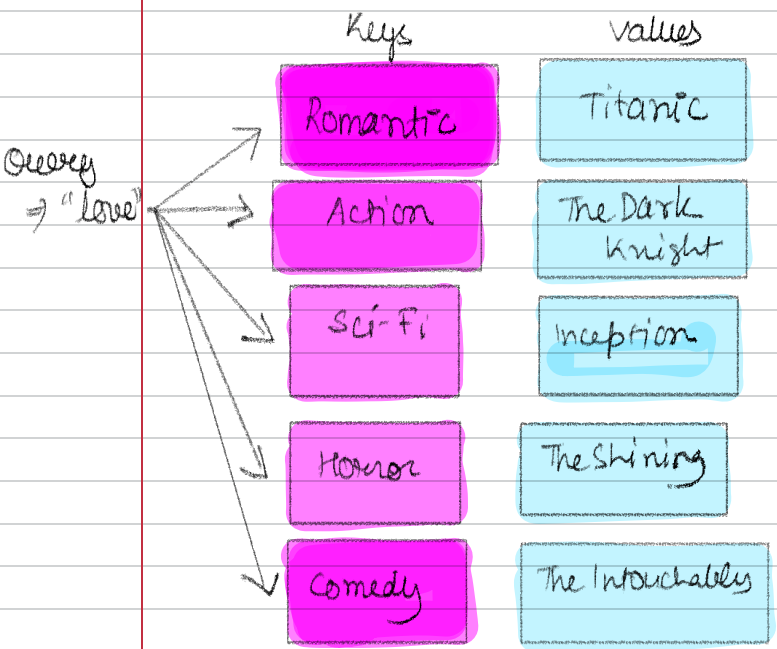
$d_k = d_{model}/h$


dmodel
dk
seq
Input
(seq, dmodel)
Q
(seq, dm)
W^Q
(dm, dm)
Q'
(seq, dm)
K
(seq, dm)
W^K
(dm, dm)
K'
(seq, dm)
V
(seq, dm)
W^V
(dm, dm)
V'
(seq, dm)
I
II
III
IV

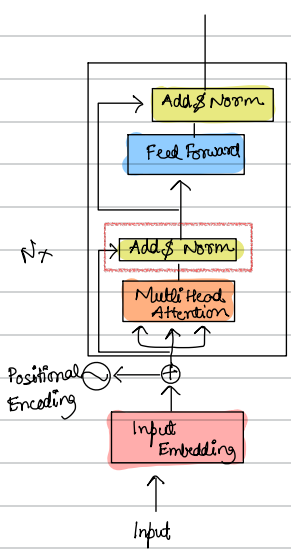

dmodel
dk
seq
Input
(seq, dmodel)
Q
(seq, dm)
W^Q
(dm, dm)
Q'
(seq, dm)
K
(seq, dm)
W^K
(dm, dm)
K'
(seq, dm)
V
(seq, dm)
W^V
(dm, dm)
V'
(seq, dm)
I
II
III
IV
HEAD I
HEAD II
HEAD III
HEAD IV
Seq
dv
dmod
Multi Head
(Q, K, V)
H
(seq, h x dv)
W^O
(h x dv, dmod)
MH - A
(seq, dmod)

Why query, key and values?

The internet says that these terms come from the database terminology or the Python-like dictionaries.

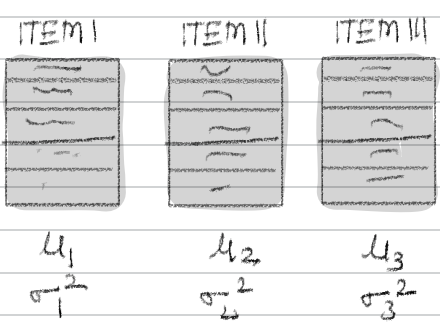


Encoders



What is Layer normalization?

Batch of 3 items

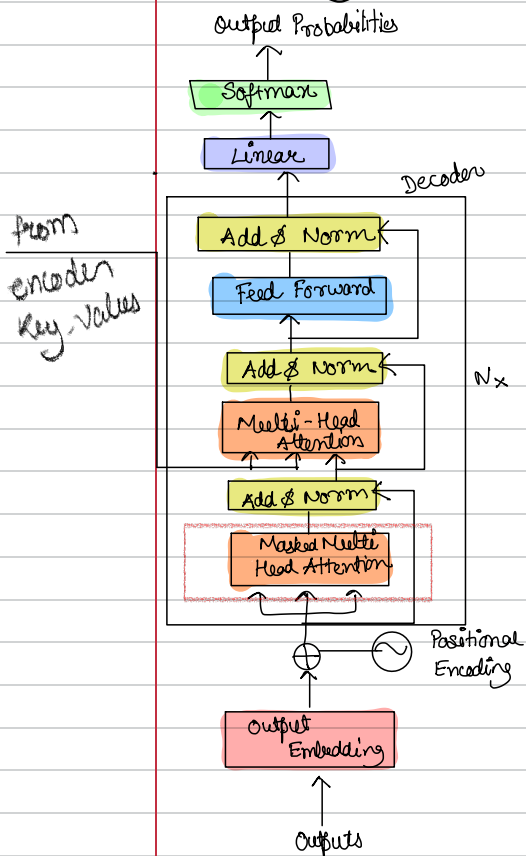


and then replace each value with

$$\hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

we also introduce two parameters usually called gamma(γ) and beta(β) that introduce some fluctuations in the data, because having all the values b/w 0 and 1 may be too restrictive for the network.

DECODER



What is Masked Multi-Head Attention ?

Our goal is to make the model causal: it means the output at a certain position can only depend on the words on the previous position.

THE MODEL MUST NOT BE ABLE TO SEE FUTURE WORDS.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	~~0.119~~	~~0.154~~	~~0.105~~	~~0.154~~	~~0.152~~
CAT	0.124	0.271	0.201	0.128	0.154	0.[illegible]
IS	0.147	0.132	0.262	~~0.117~~	~~0.208~~	~~0.135~~
A	0.210	0.128	0.206	0.212	~~0.119~~	~~0.125~~
LOVELY	0.146	0.158	0.152	0.143	0.227	~~0.124~~
CAT	0.195	0.114	0.203	0.103	0.157	0.229

6X6

I want the words to only see the words before them.

Replace them with $-\infty$

INFERENCE AND TRAINING OF A TRANSFORMER

Training

I love you very much ⟶ Ti amo molto