Transformers Explained

→ Self - Attention Mechanism

By: Kanav Bansal
(ThatAIGuy)

By: Kanav Bansal
(That AI Guy)

Summary ↴

2014 → Sequence to Sequence Learning with Neural Network
Proposed - Encoder Decoder architecture

2015 → Neural Machine Translation with Joint Learning to align & Translate
Proposed - Bahdanau Attention Mechanism

2017 → Attention is all you need
Proposed - Transformer Architecture with Self-Attention

2018 → Universal Language Model Fine-Tuning for Text Classification
Proposed - Language Modeling as pre-training Task

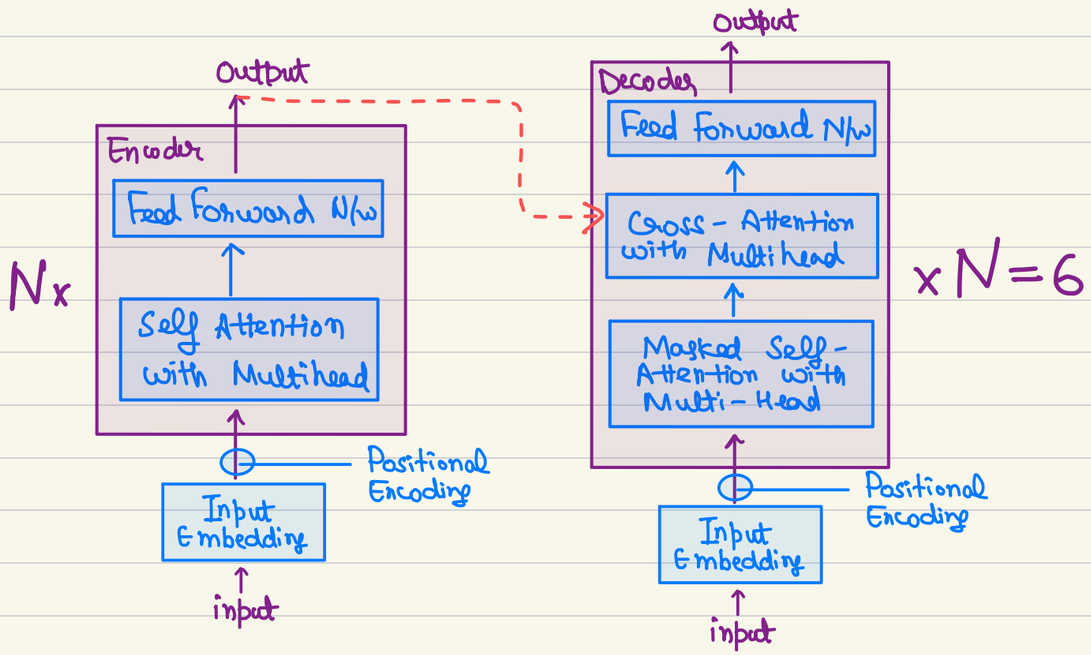
By: Kanav Bansal
(That AI Guy)

Paper: Attention is all you need

2017 → Proposed Transformer Architecture which used:
- → Encoder Decoder Architecture
- → Self-Attention Mechanism

BUILDING BLOCKS

(Note: This is a simplified view of Transformers)



ENCODER BUILDING BLOCK

1. Positional Embedding
   - Adds the notion of Seq info
2. Self-Attention with Multihead
   - Attention replaced need of recurrence & makes the process of embedding calculation parallel
   - Attention mechanism will find how each word relates to all other words in a sequence & generates Contextual Embeddings

   Ques: How to find word similarities?
   - Attention will run dot products b/w word vectors & determine the strongest relationships of a word with all other words

   Ques: How to speed up the calculations?
   - For each attention sublayer, the original transformer model runs not only one but eight attention mechanisms in parallel to speed up the calculations. This process is done using 'multi-head attention'.
3. Fully Connected Positionwise Feed Forward N/w
   - Improves the word association by applying non-linear transformations.

By: Kanav Bansal
(That AI Guy)

TRANSFORMER'S

↓

Attention is all you need (2017 Paper)

Why CELEB STATUS ?

1. Scalable & Parallel Training
2. Revolutionized NLP with LLMs
3. Unified DL Approaches for text, image, audio & video data
4. Multi-Modal
5. Accelerated Gen AI

Encoder

Has self-Attention with multi-head

&

Feed forward ANN

↑

Great at understanding Text

Eg: BERT

Decoder

Mask Self-Attention & Cross-Attention with Multi-head

&

feed forward ANN

↑

Great at Generating Text

Eg: GPT-1

By: Kanav Bansal
(That AI Guy)

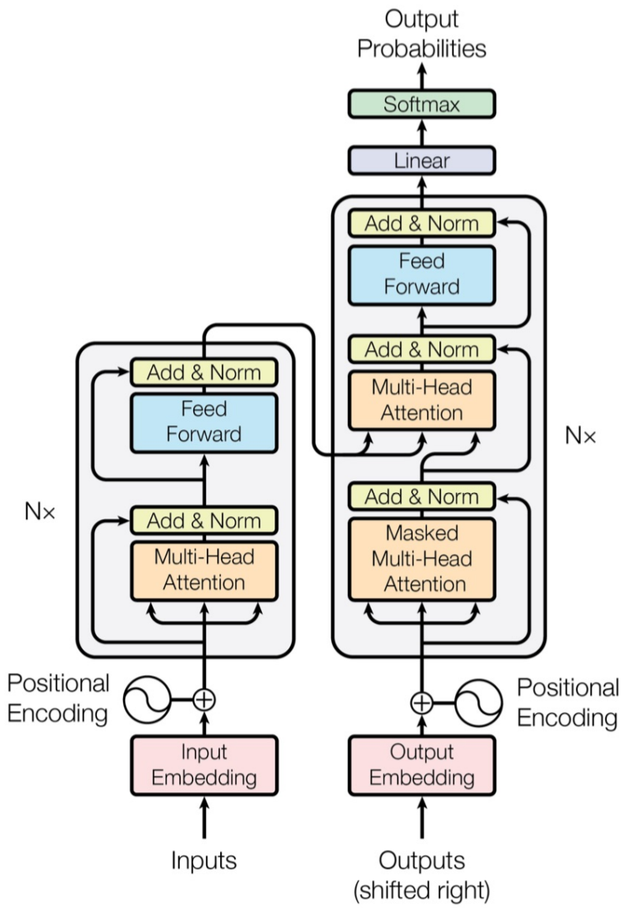


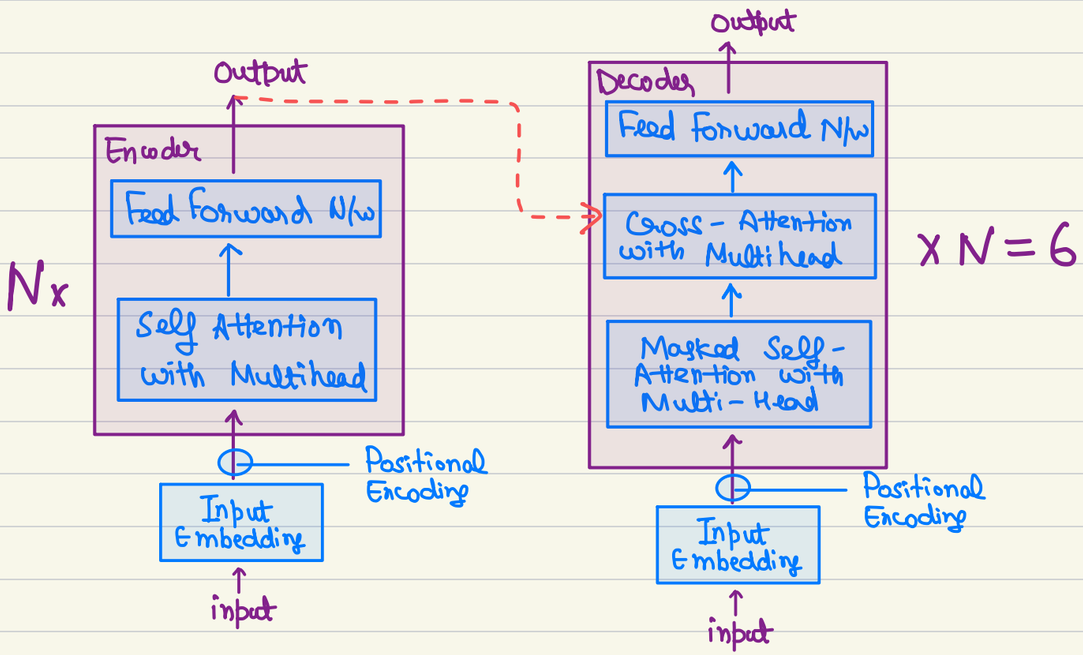
Figure 1: The Transformer - model architecture.

Advantages
- Parallel & Scalable
- Multi Modal input & output

Disadvantages
- Huge compute resources
- Huge amount of data
- Overfit
- Energy

BUILDING BLOCKS

By: Kanav Bansal
(That AI Guy)

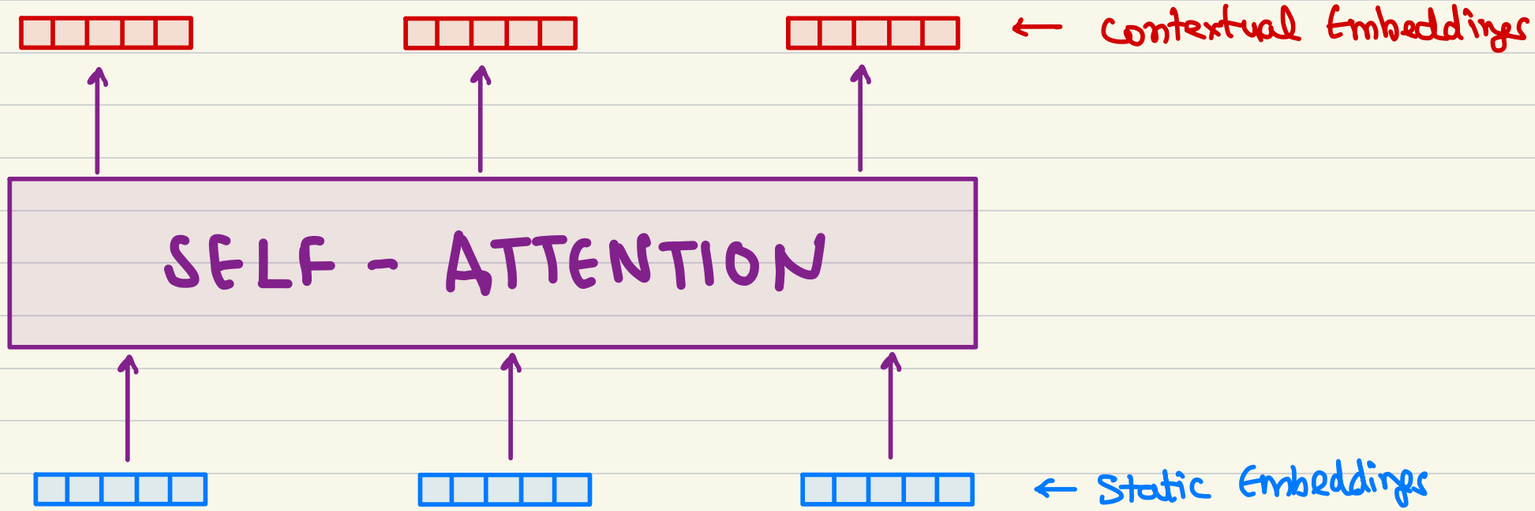


Encoder is good at understanding text

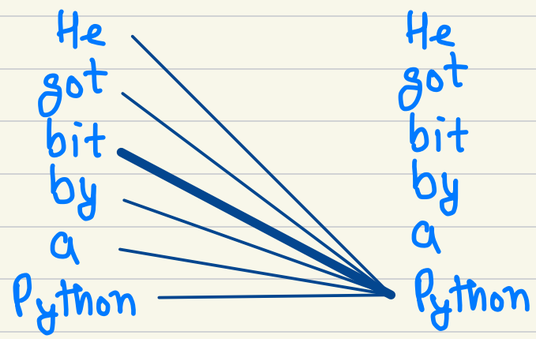
Decoder is good at generating text

UNDERSTANDING SELF-ATTENTION ↴

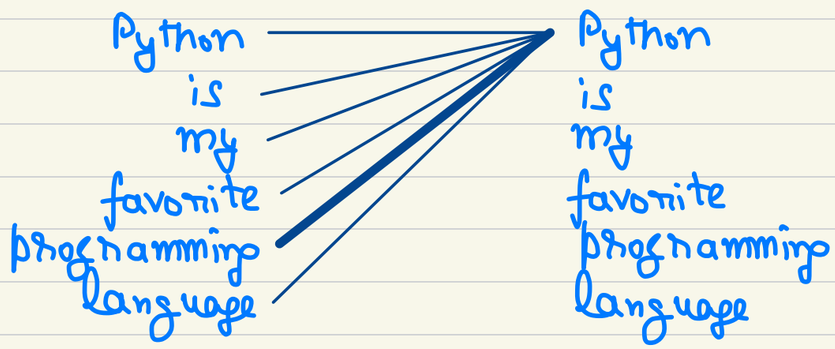
By: Kanav Bansal
(That AI Guy)



Ex: Sent A → He got bit by a Python



Sent B → Python is my favorite programming language



By: Kanav Bansal
(That AI Guy)

Example

Sentence 1: Python executes quickly

Sentence 2: Python crawls quickly

Python = 0.5 Python + 0.3 executes + 0.2 quickly

executes = 0.2 Python + 0.7 executes + 0.1 quickly

quickly = 0.1 Python + 0.1 executes + 0.8 quickly

Python = 0.4 Python + 0.4 crawls + 0.2 quickly

crawls = 0.3 Python + 0.6 crawls + 0.1 quickly

quickly = 0.1 Python + 0.1 crawls + 0.8 quickly

$e_{Python} \cdot e_{Python}$ $e_{Python} \cdot e_{executes}$ $e_{Python} \cdot e_{quickly}$

$z_{Python} = \boxed{0.5}\, e_{Python} + \boxed{0.3}\, e_{executes} + \boxed{0.2}\, e_{quickly}$

$z_{executes} = 0.2\, e_{Python} + 0.7\, e_{executes} + 0.1\, e_{quickly}$

$z_{quickly} = \boxed{0.1}\, e_{Python} + 0.1\, e_{executes} + 0.8\, e_{quickly}$

$e_{quickly} \cdot e_{Python}$

$e_{Python} \cdot e_{crawls}$

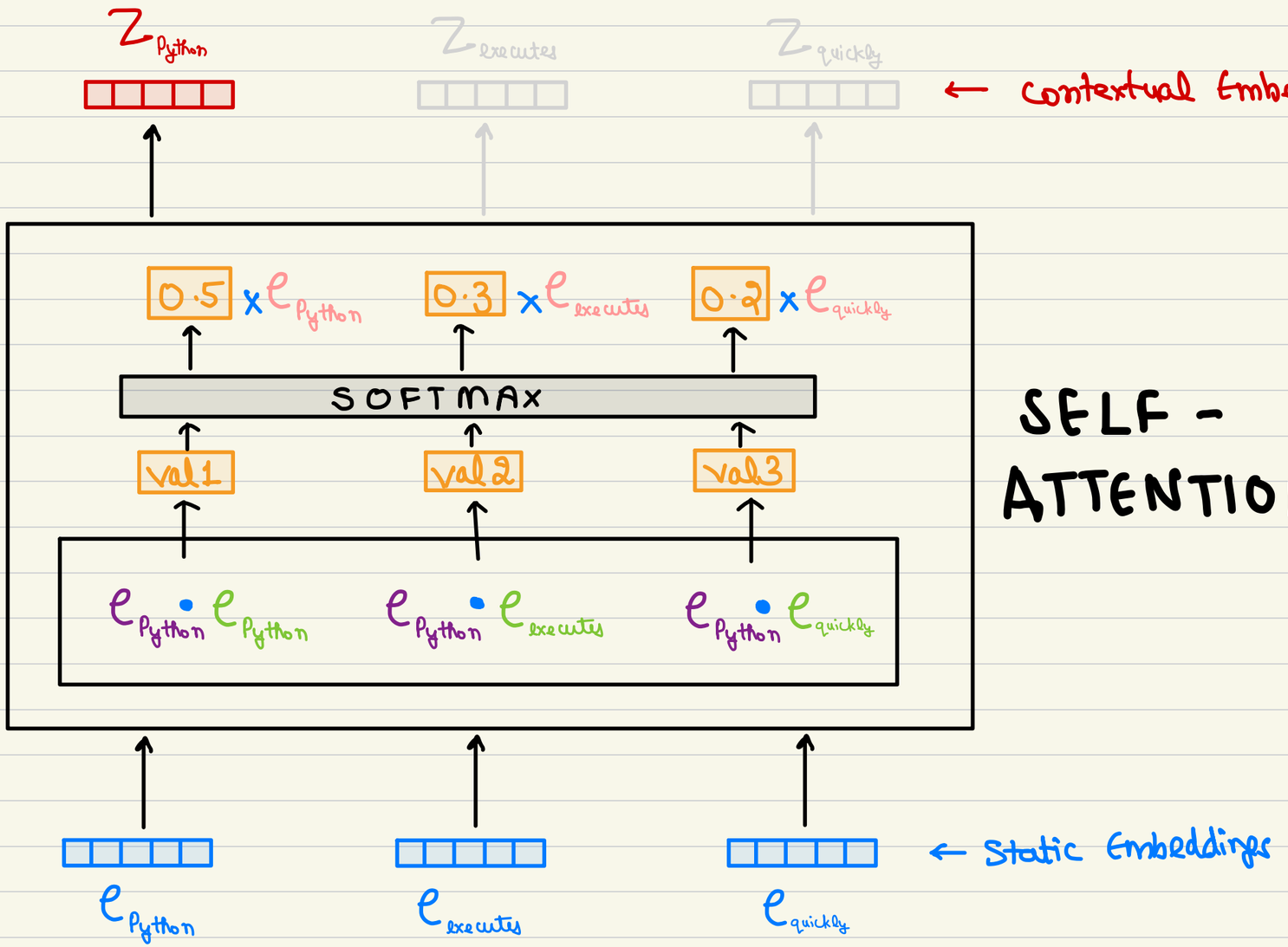
$z_{Python} = 0.4\, e_{Python} + \boxed{0.4}\, e_{crawls} + 0.2\, e_{quickly}$

$z_{crawls} = 0.3\, e_{Python} + 0.6\, e_{crawls} + 0.1\, e_{quickly}$

$z_{quickly} = 0.1\, e_{Python} + 0.1\, e_{crawls} + 0.8\, e_{quickly}$

By: Kanav Bansal
(That AI Guy)

The complete process → (Calculation for the "Python" Query)



SELF-ATTENTION

SELF ATTENTION

By: Kanav Bansal
(That AI Guy)

1. init the 3 weight matrix

These weights get updated at training time

W_q W_K W_V

512 ↓

Shape = Dxd

↑ 64

2. Calc 1 :

$Q = X W_q$ $K = X W_K$ $V = X W_V$

$[Q]_{Txd}$ $[K]_{Txd}$ $[V]_{Txd}$

Shape = (TxD)x(Dxd)
= Txd

3. Calc 2:

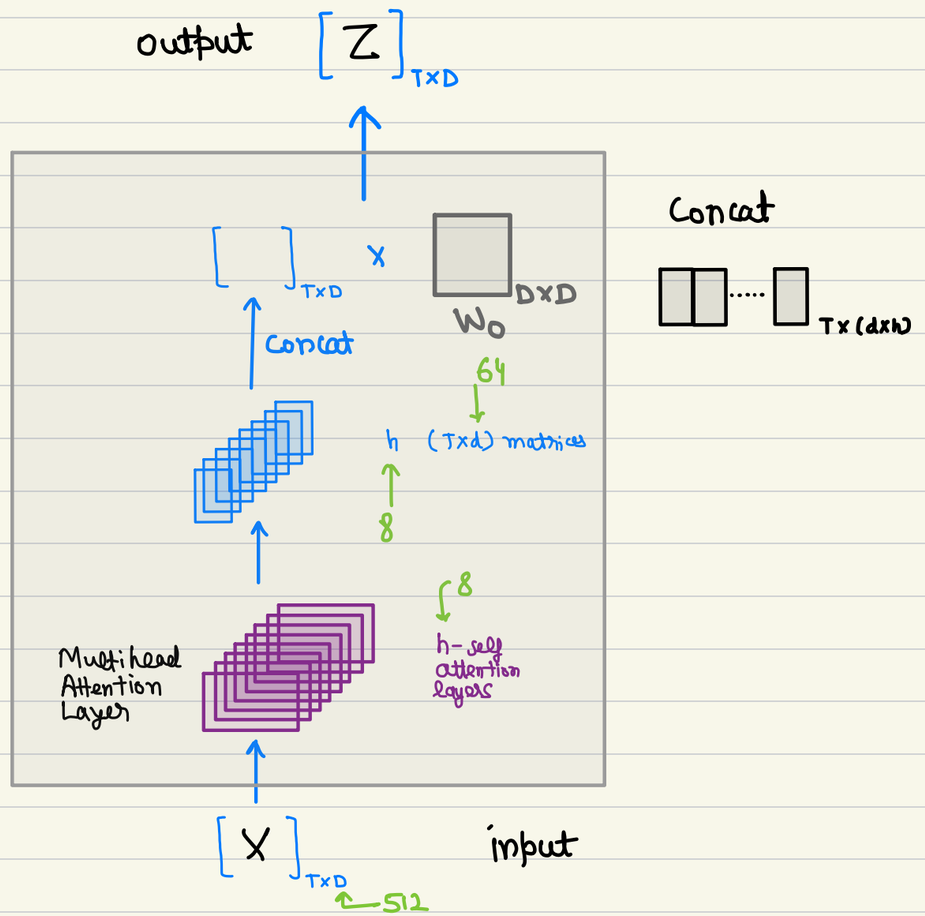
$$\text{Softmax}\left(\frac{Q K^T}{\sqrt{d}}\right) V$$

This returns a matrix of shape TxT.
Intuitively, we can think of this matrix as the importance of each word on another.
AKA Attention Score

Shape = ((Txd)x(dxT))x(Txd)
= (TxT)x(Txd)
= (Txd)

MULTI-HEAD SELF-ATTENTION

As it uses multiple layers of self-attention in parallel

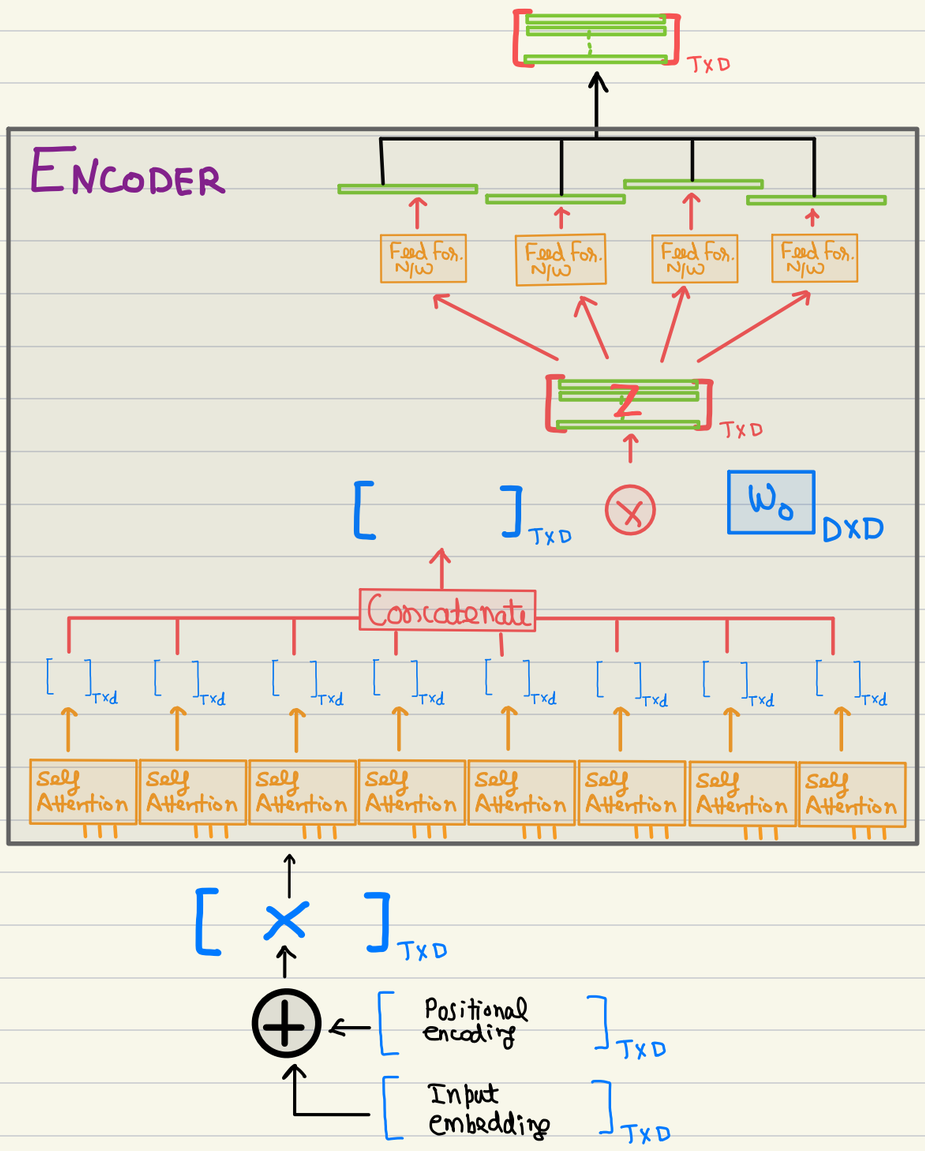


Concat

Tx(dxh)

ENCODER

By: Kanav Bansal
(That AI Guy)



$$PE_{(pos,2i)} = sin(pos/10000^{2i/D})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/D})$$