



Transformers

Rowel Atienza, Ph.D.

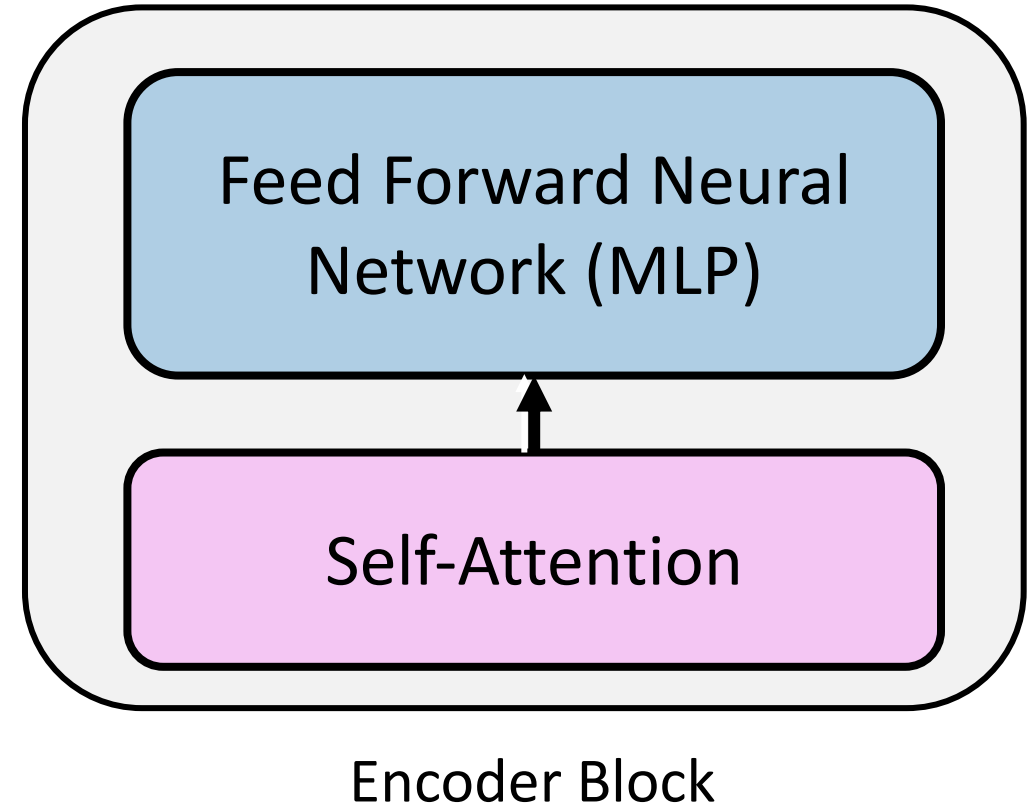
University of the Philippines

github.com/roatienza

2022

Transformer Encoder/Decoder Unit Details

Operations: Linear, Layer Norm, Activation, Tensor Multiply/Add, Softmax



Types of data that transformers can process

Any

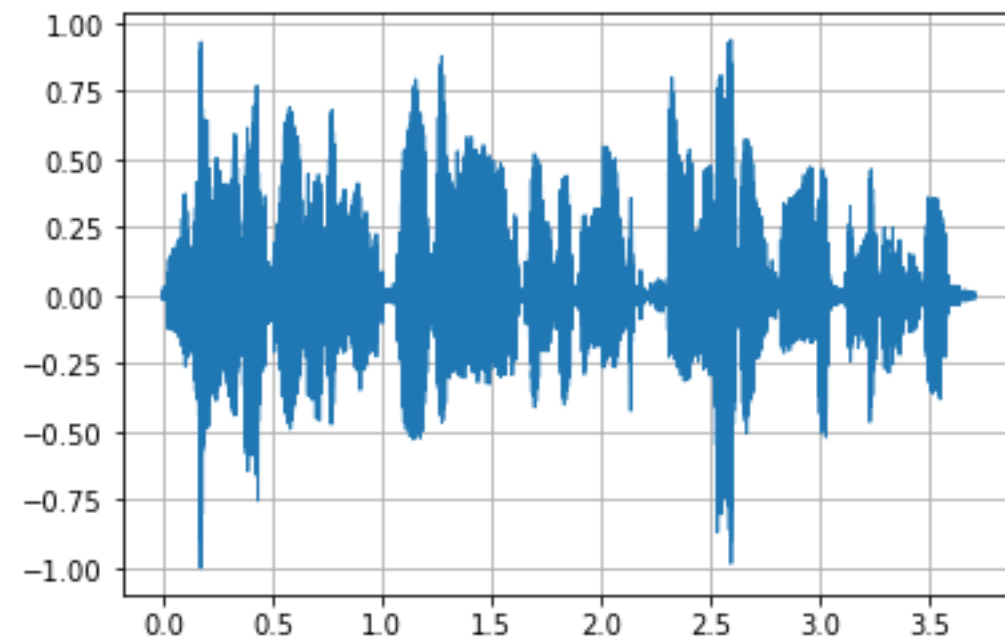


WIKIPEDIA
The Free Encyclopedia

COCO 2017 Keypoint Detection Task



Waveform



Input Embedding is an $n - dim$ vector

x_1^T

.1	-2	.4	-1
----	----	----	----

cat

x_2^T

.3	1	-1	2
----	---	----	---

opened

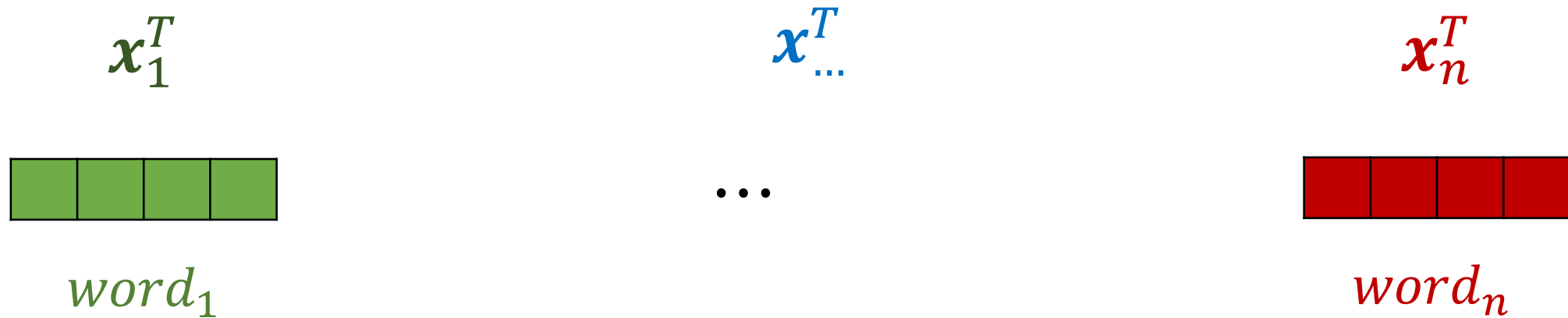
x_3^T

.1	.0	1	-1
----	----	---	----

its

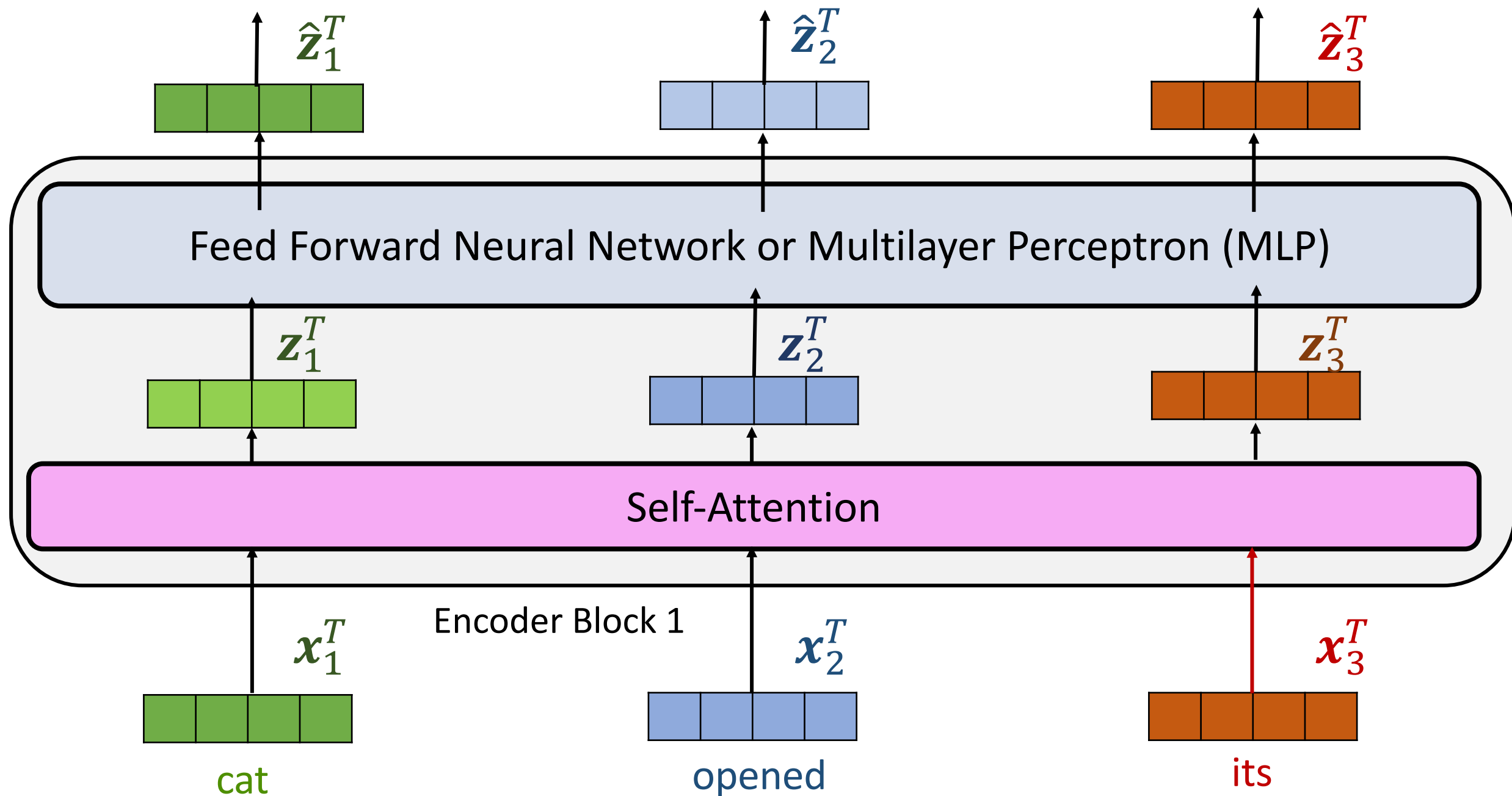
Example: Each word is converted into a 512-dim embedding vector.
In the simple example above, it is 4-dim.

The Length of the Input is n

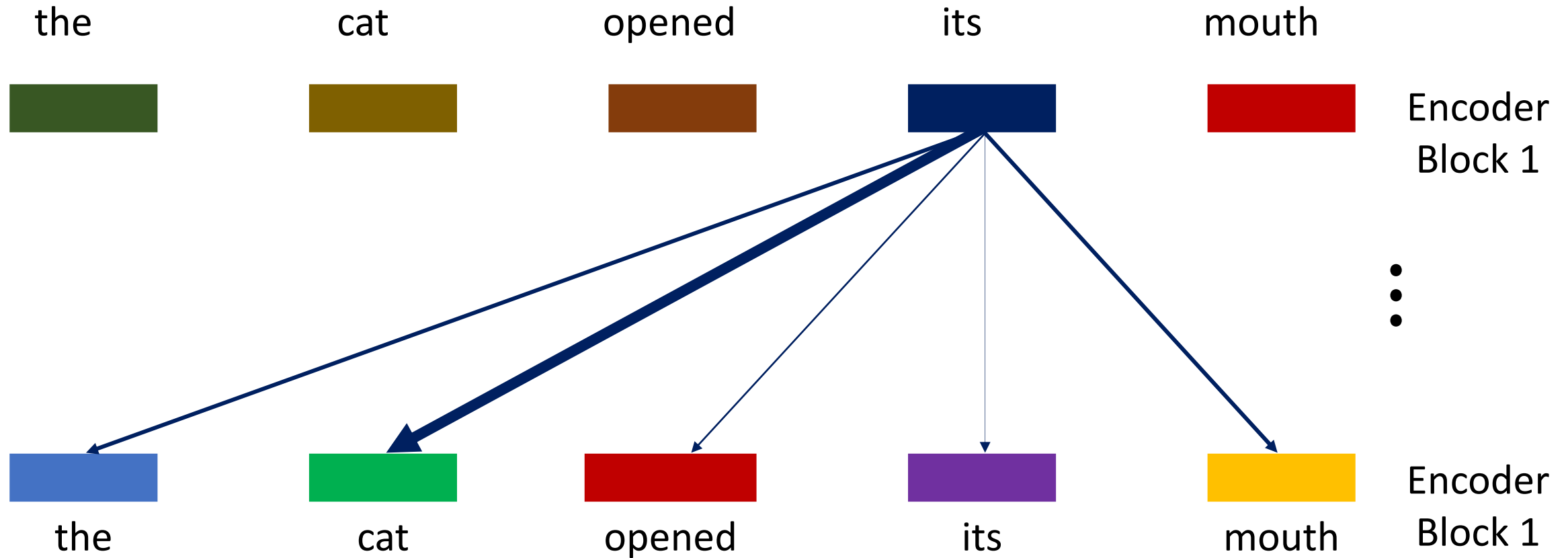


Example: n could be the maximum possible length of a sentence.

Encoder with Latent Variables \mathbf{z}_i



Attention between 2 words



Attention as measured by the width of the arrow

Query *Key* *Value*

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

d_k is keys/queries dim (e.g. 4)

$$\textit{Attention} = \textit{softmax}\left(\frac{\begin{array}{|c|c|} \hline \begin{array}{|c|c|c|c|} \hline \text{purple grid} \\ \hline \end{array} & \begin{array}{|c|c|c|c|} \hline \text{yellow grid} \\ \hline \end{array} \\ \hline \end{array}^T}{\sqrt{d_k}}\right) \begin{array}{|c|c|c|c|} \hline \text{magenta grid} \\ \hline \end{array}$$

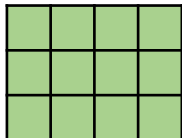
$$\textit{Attention}(Q, K, V) = Z = \begin{array}{|c|c|c|c|} \hline \text{green grid} \\ \hline \end{array}$$

Values With all things considered, this is where the attention should be

Keys What I think the features should be

Queries What I think the features should be

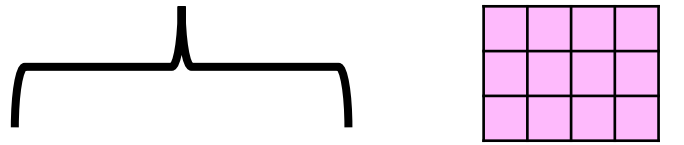
$$\text{Attention} = \text{softmax} \left(\frac{\begin{matrix} \text{Query Matrix} & \text{Key Matrix}^T \end{matrix}}{\sqrt{d_k}} \right)$$

$\text{Attention}(Q, K, V) = Z =$ 

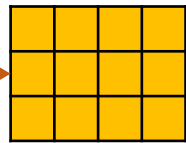
The diagram illustrates the attention mechanism. It shows three input matrices: a purple 4x4 Query Matrix, a yellow 4x4 Key Matrix, and a pink 4x4 Value Matrix. The Query and Key matrices are combined in a dot product (indicated by the superscript T on the Key matrix) and then divided by the square root of the key dimension d_k . The result of this operation is passed through a softmax function to produce the attention matrix Z, which is a green 4x4 grid. Arrows indicate the flow of information from the input matrices to the final attention matrix Z.

Consider an Attention Layer Examining a Digit

I can see everything that you can see. You are a part of digit 3.

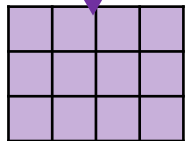


Values



Keys

I saw a bar to the left. I think I am a part of digit 3, 5, 6, or 8.



Queries

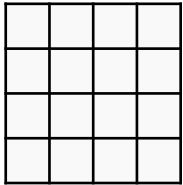
I think I am a part of digit 3. 5 is also possible.

Example: Let us focus on the lower-right patch only

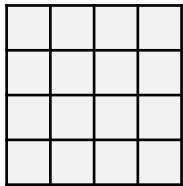
Self-Attention

Attention Layer 1 Learnable Parameters

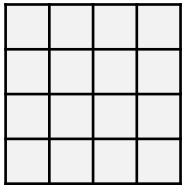
W^Q



W^K



W^V



Embedding

cat



x_1^T

opened

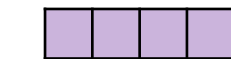


x_2^T

its



x_3^T



$q_1^T = x_1^T W^Q$



$q_2^T = x_2^T W^Q$



$q_3^T = x_3^T W^Q$



$k_1^T = x_1^T W^K$



$k_2^T = x_2^T W^K$



$k_3^T = x_3^T W^K$



$v_1^T = x_1^T W^V$



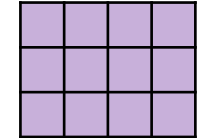
$v_2^T = x_2^T W^V$



$v_3^T = x_3^T W^V$

$X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix}$ Encoder 1 Inputs

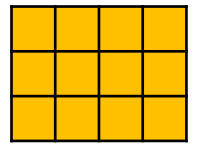
Queries



Q

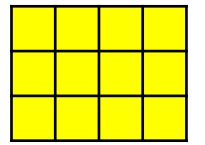
$Q = XW^Q$

Keys



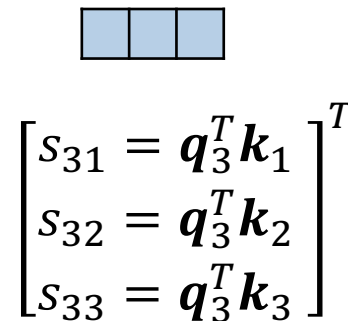
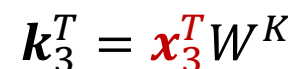
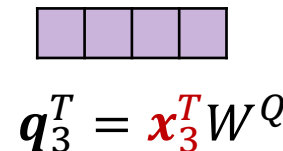
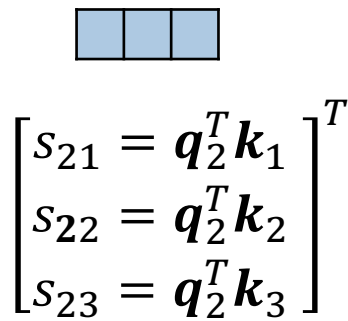
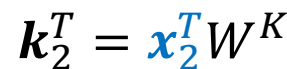
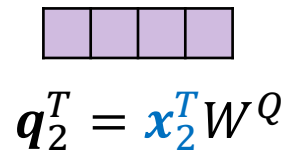
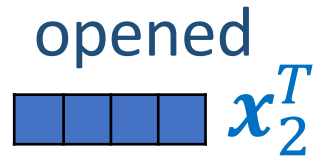
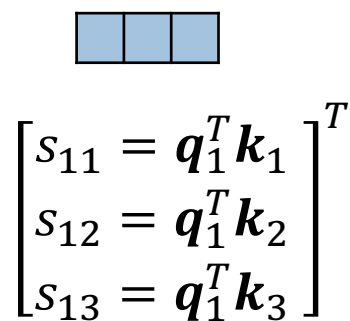
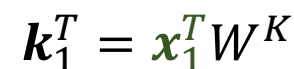
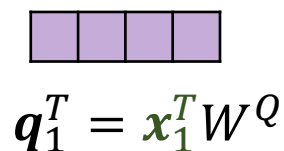
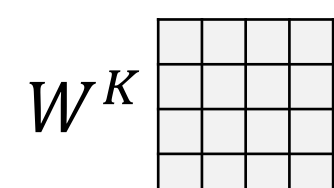
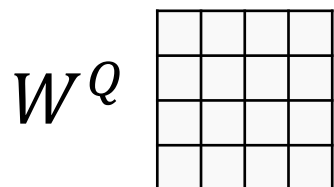
$K = XW^K$

Values

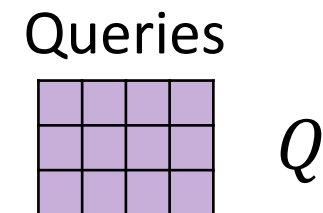


V

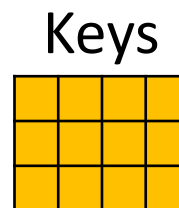
$V = XW^V$



$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix}$ Encoder 1 Inputs



$Q = XW^Q$

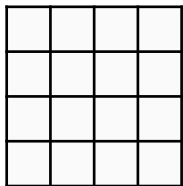


$K = XW^K$

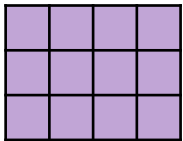
Scores $\begin{bmatrix} \text{Queries} \end{bmatrix} \left(\begin{bmatrix} \text{Keys} \end{bmatrix} \right)^T = \begin{bmatrix} \text{Scores} \end{bmatrix} = S$

Multi-Head
(eg 8-head)

$$W_1^Q$$

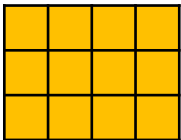


Queries



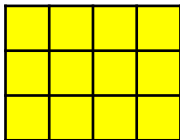
$$Q_1 = XW_1^Q$$

Keys



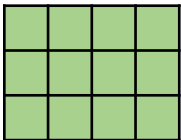
$$K_1 = XW_1^K$$

Values



$$V_1 = XW_1^V$$

Head 1: Z_1



opened x_2^T

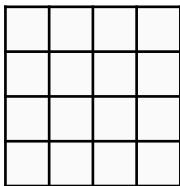


its x_3^T

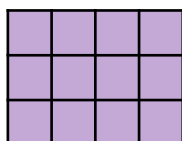


$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} \text{ Encoder 1 Inputs}$$

$$W_8^Q$$

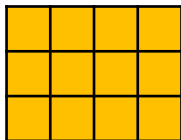


Queries



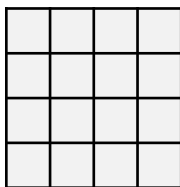
$$Q_8 = XW_8^Q$$

Keys

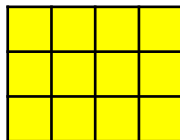


$$K_8 = XW_8^K$$

$$W_8^K$$

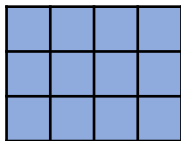


Values



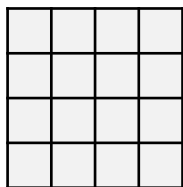
$$V_8 = XW_8^V$$

Head 8: Z_8

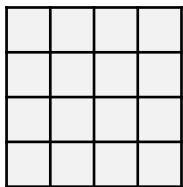


...

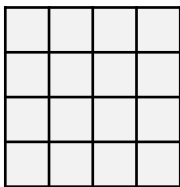
$$W_1^K$$



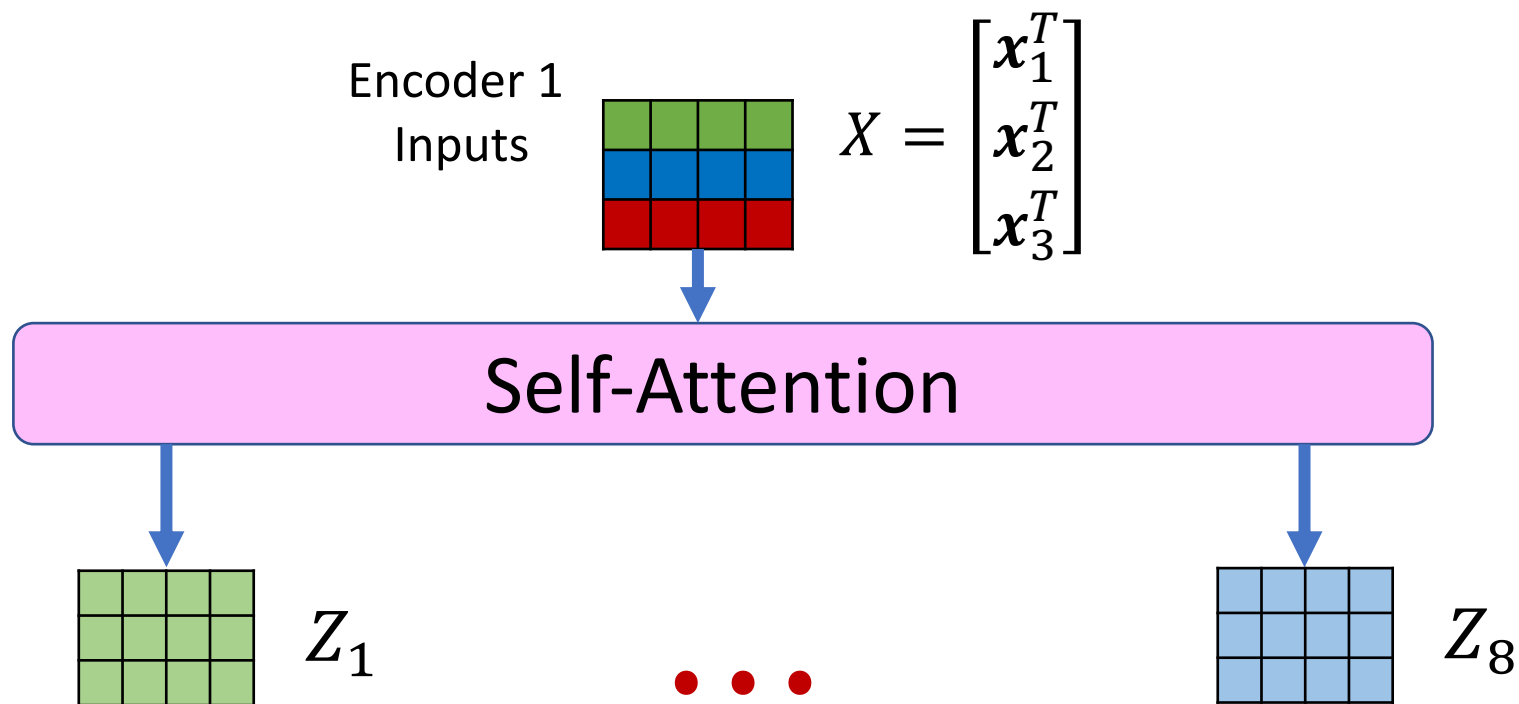
$$W_1^V$$



$$W_8^V$$



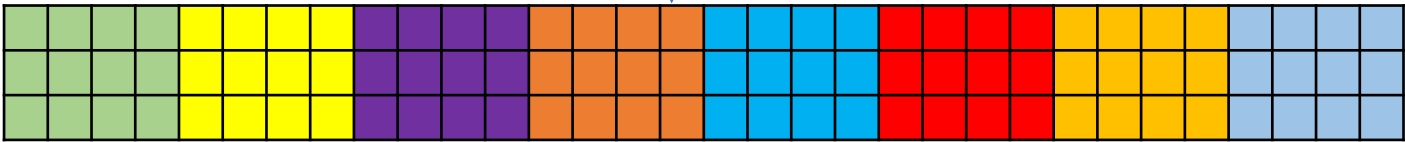
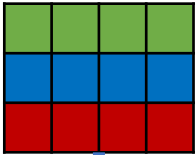
Multi-Head
(eg 8-head)



Multi-Head
(eg 8-head)
Merge Outputs
Apply Weights

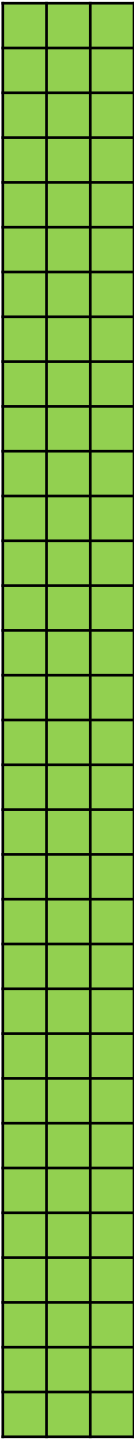
Encoder 1
Inputs

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix}$$



$$cat(Z_1, \dots, Z_8)$$

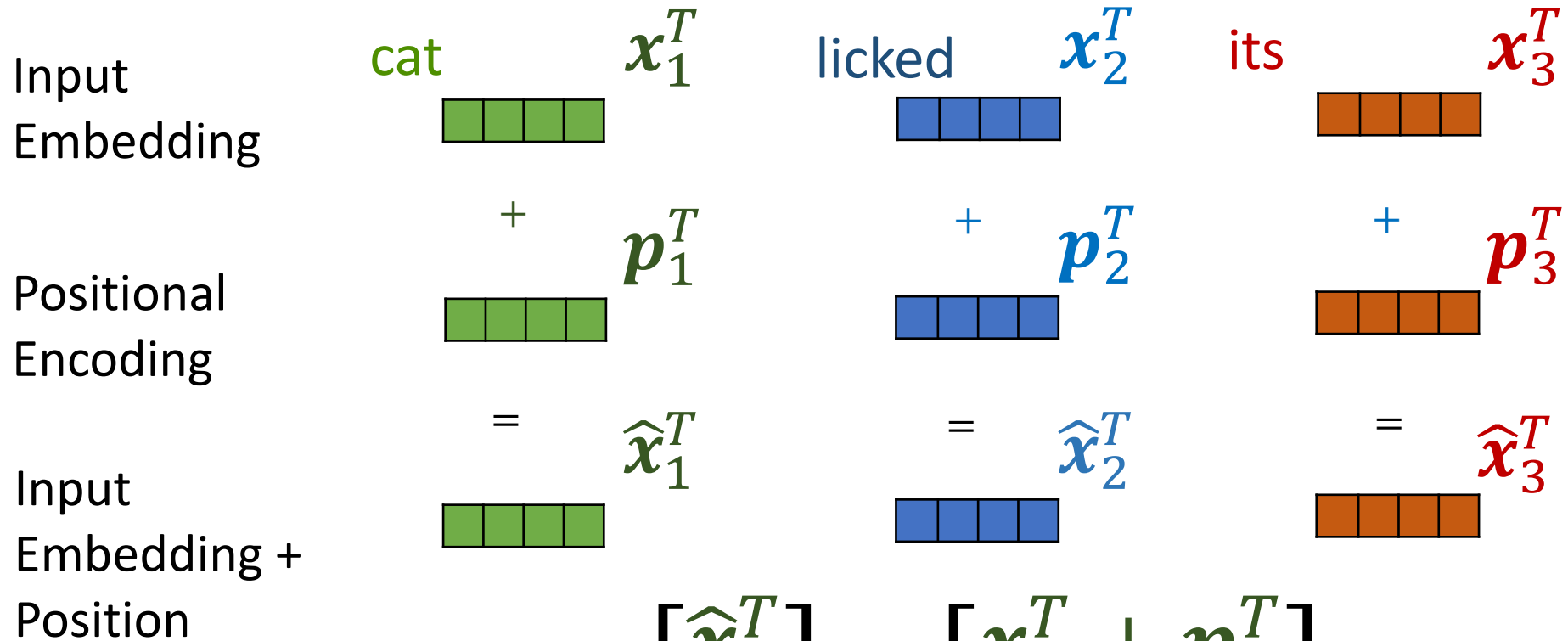
×



$$W^0$$

$$= Z$$
A 3x3 grid of yellow squares.

Adding Position Info to Inputs



Positional Encoding

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_k}}}\right) \quad dim = 2i \text{ is even}$$

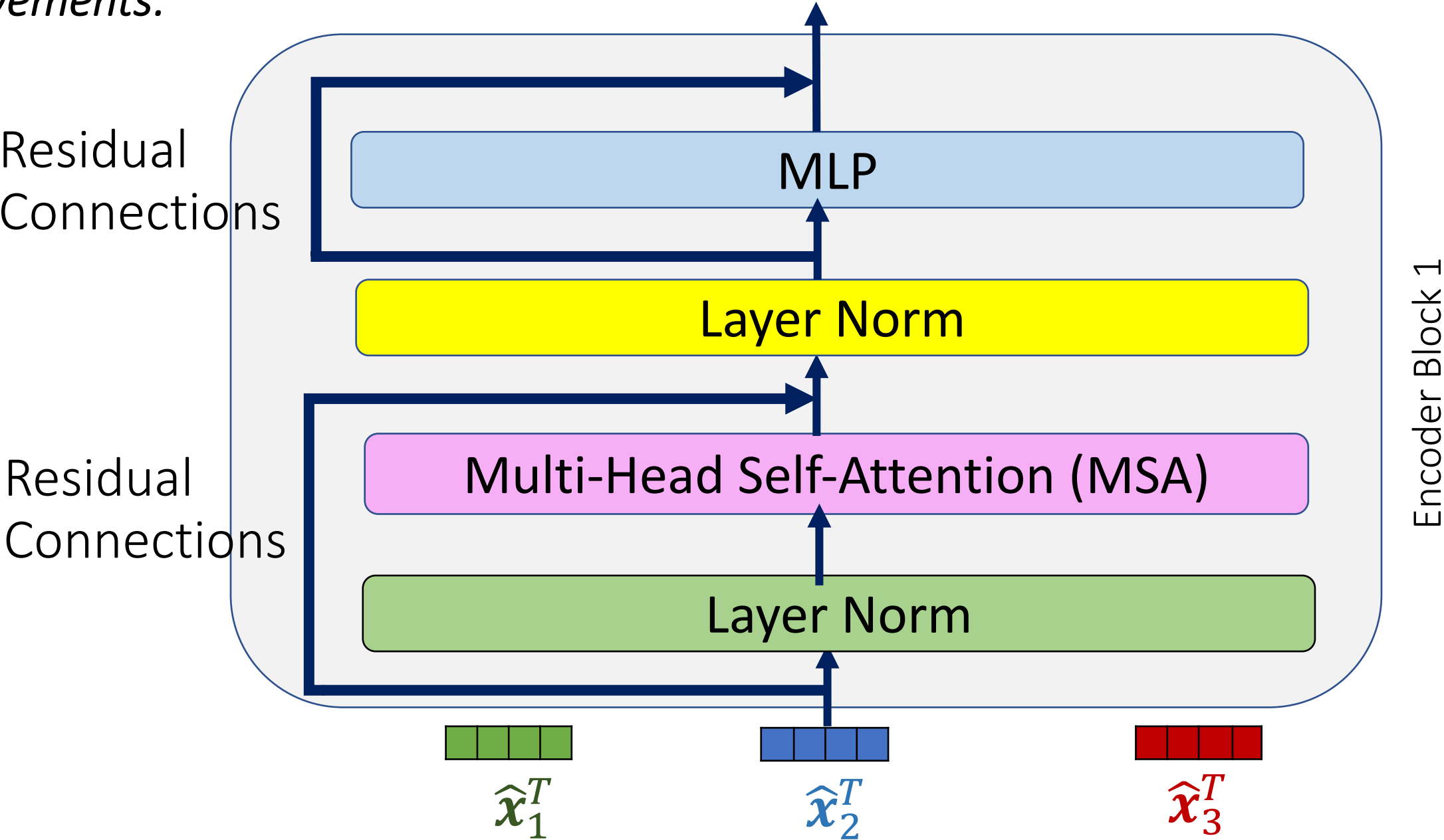
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_k}}}\right) \quad dim = 2i + 1 \text{ is odd}$$

$$pos = 0, 1, \dots, n_{pos}-1$$

$$dim = 0, 1, \dots, n_{dim}-1$$

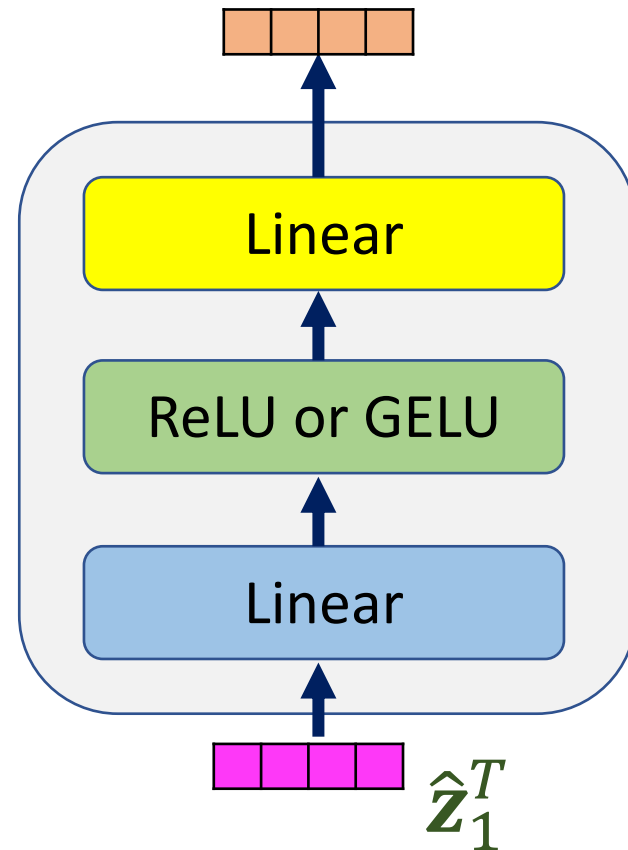
Other positional encoding methods: learnable

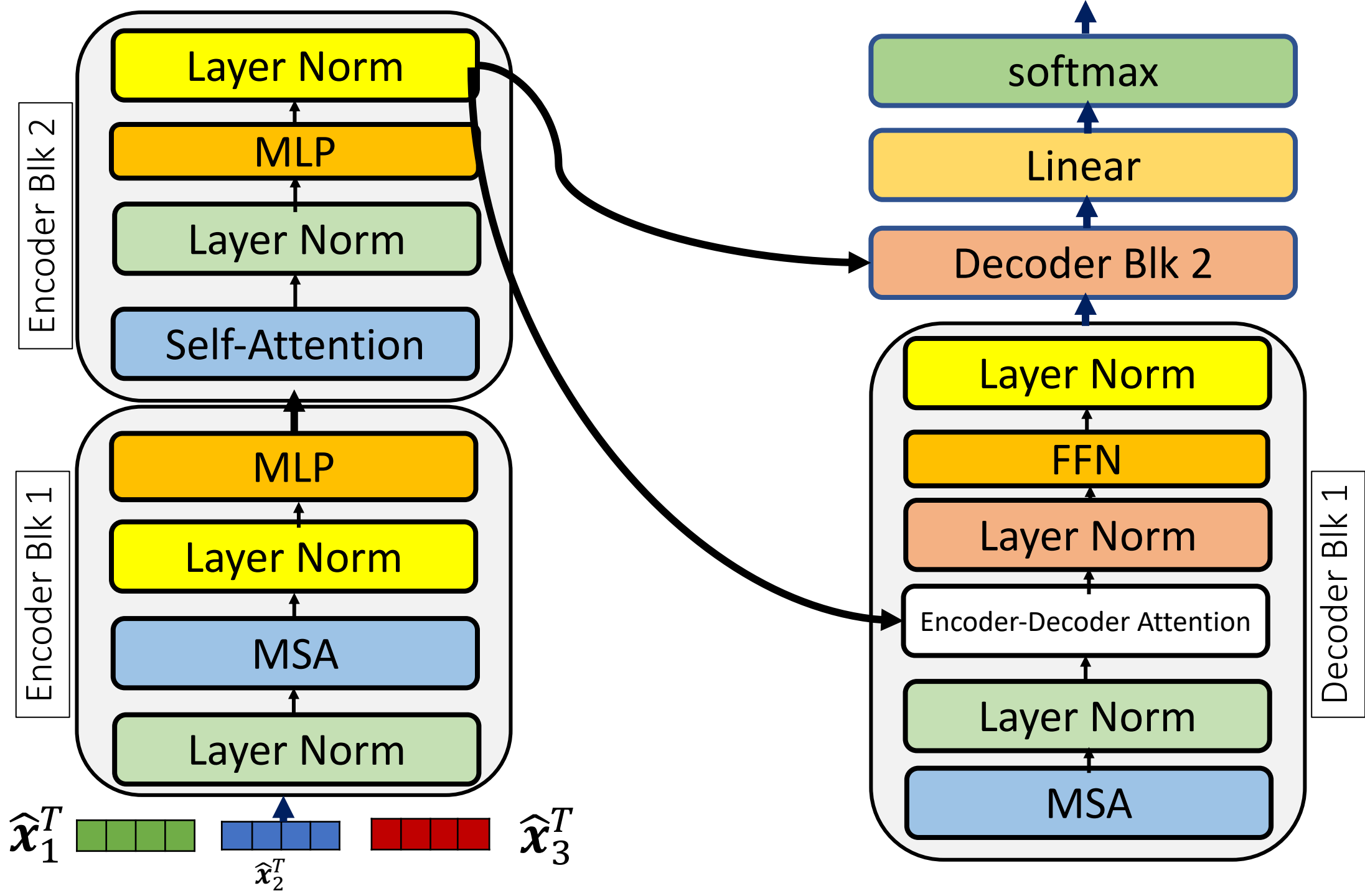
Improvements:

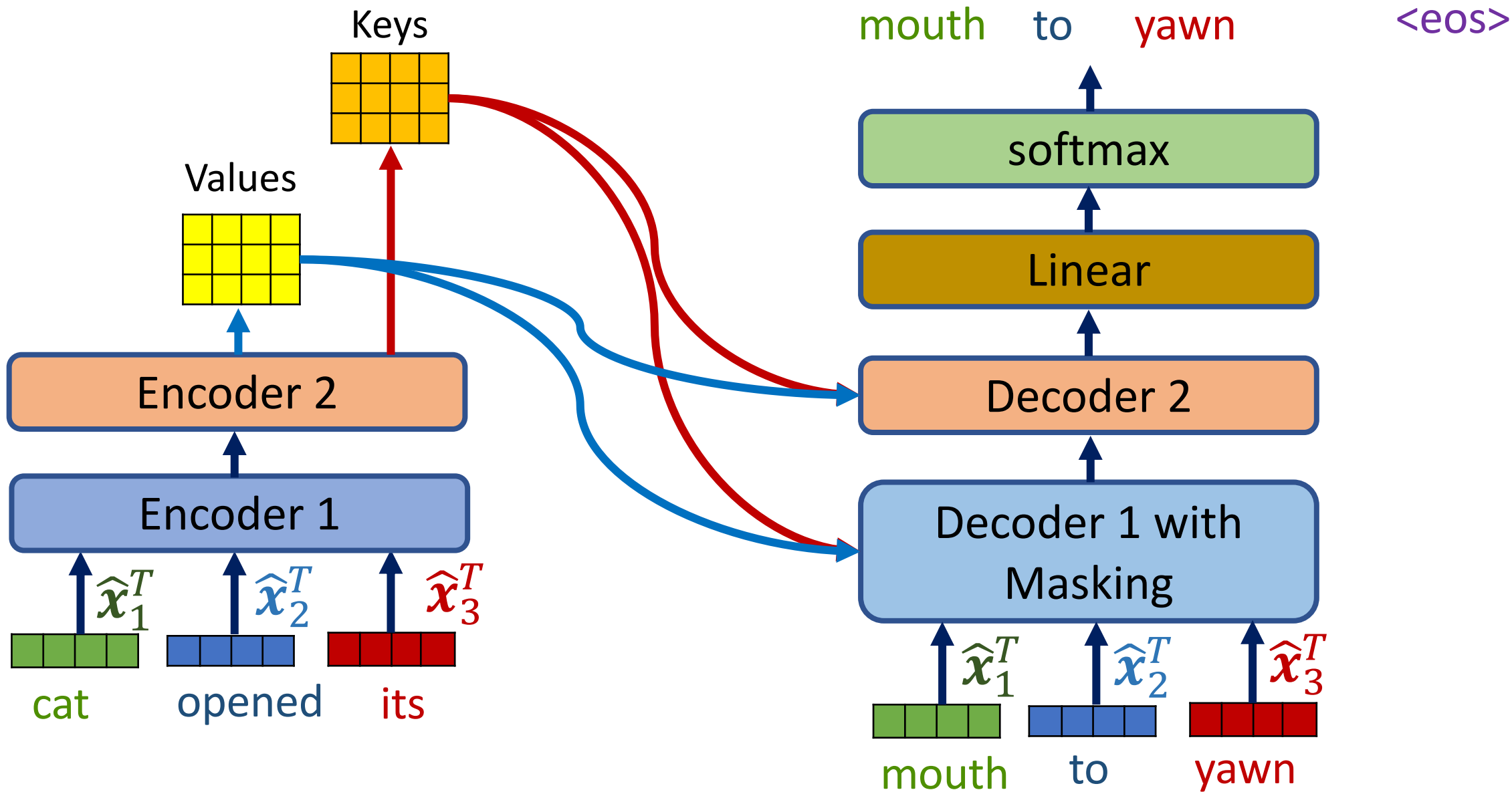


FFN: Feed Forward Neural Network (MLP)

$$MLP(x) = \max(0, xW_1 + b_1)W_2 + b_2$$







Masking prevents Decoder 1 from seeing the future. Decoder 1 relies only on the previous outputs.

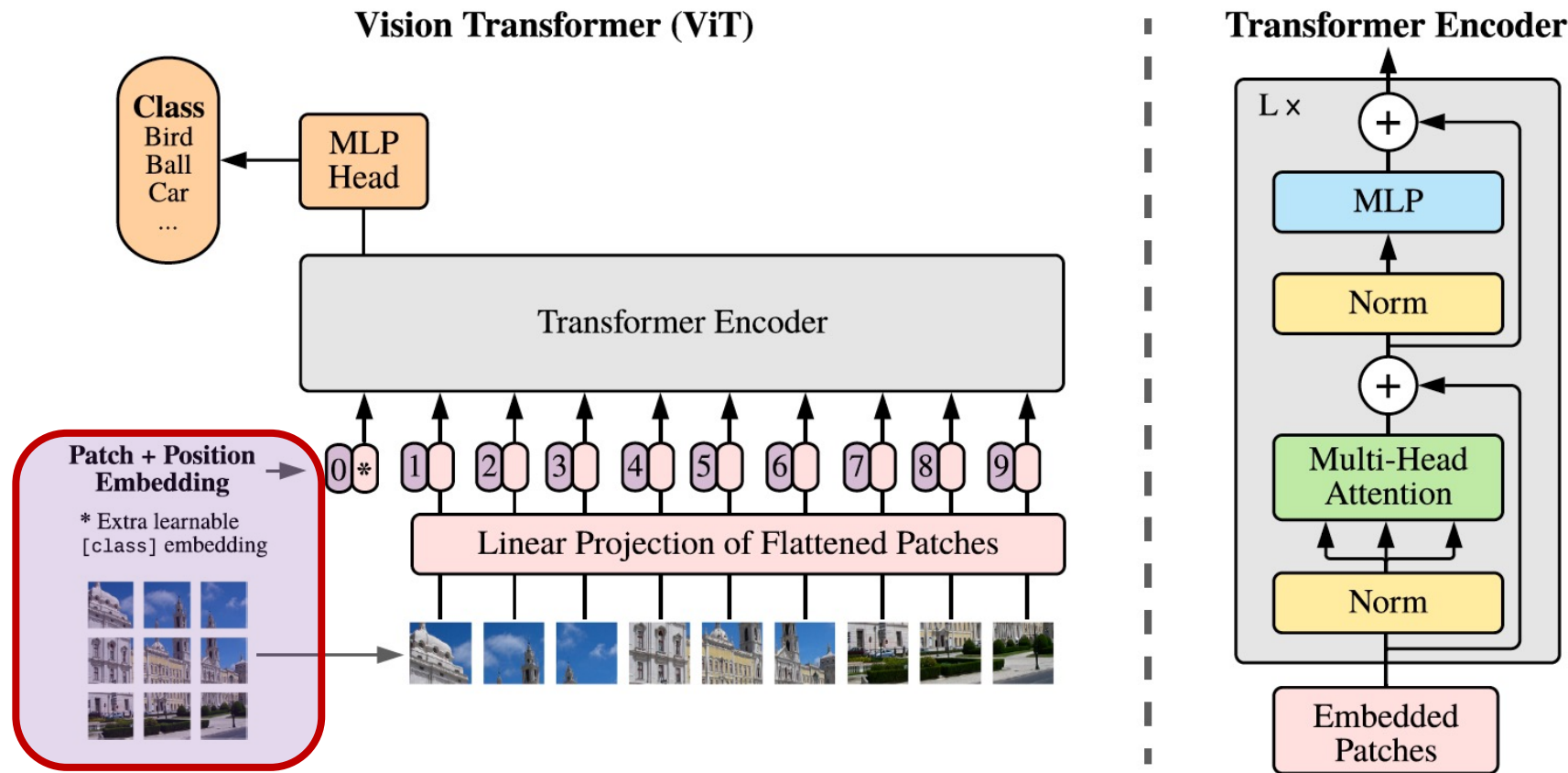


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings to the resulting sequence of vectors, and feed the patches to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, ICLR 202

Inductive Bias

Transformers lack some inductive biases inherent to CNNs, such as translation equivariance and scale invariance (w/ maxpool), and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if we train the models on large datasets (14M-300M images). We find that large scale training trumps inductive bias.

References

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020). ICLR 2021.

Illustrated Transformer, <http://jalammar.github.io/illustrated-transformer/>

Transformers from Scratch, <http://peterbloem.nl/blog/transformers>

Transformer Family, <https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html>

In Summary

Transformers could be the most important breakthrough in the recent history of deep learning

Transformers have been used to produce state-of-the-art performances in language, vision, audio, and multi-modal domains

Expect more development in this field in the near future

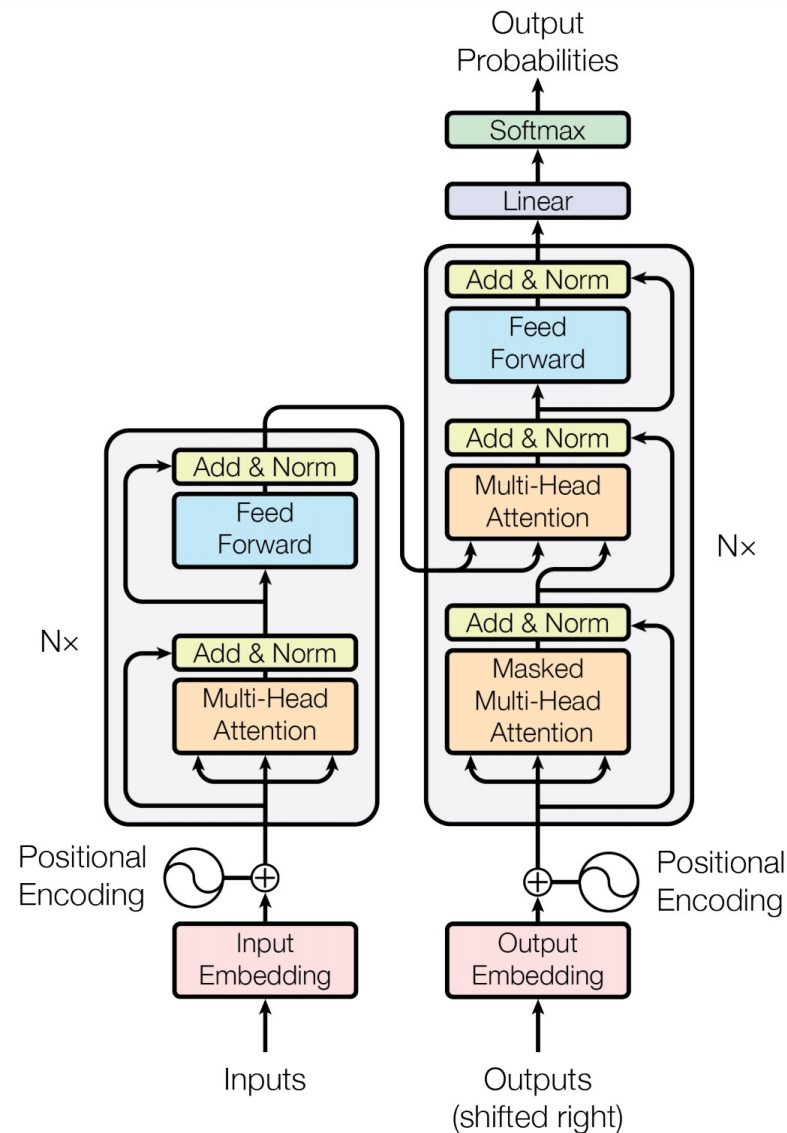


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Code demo is next