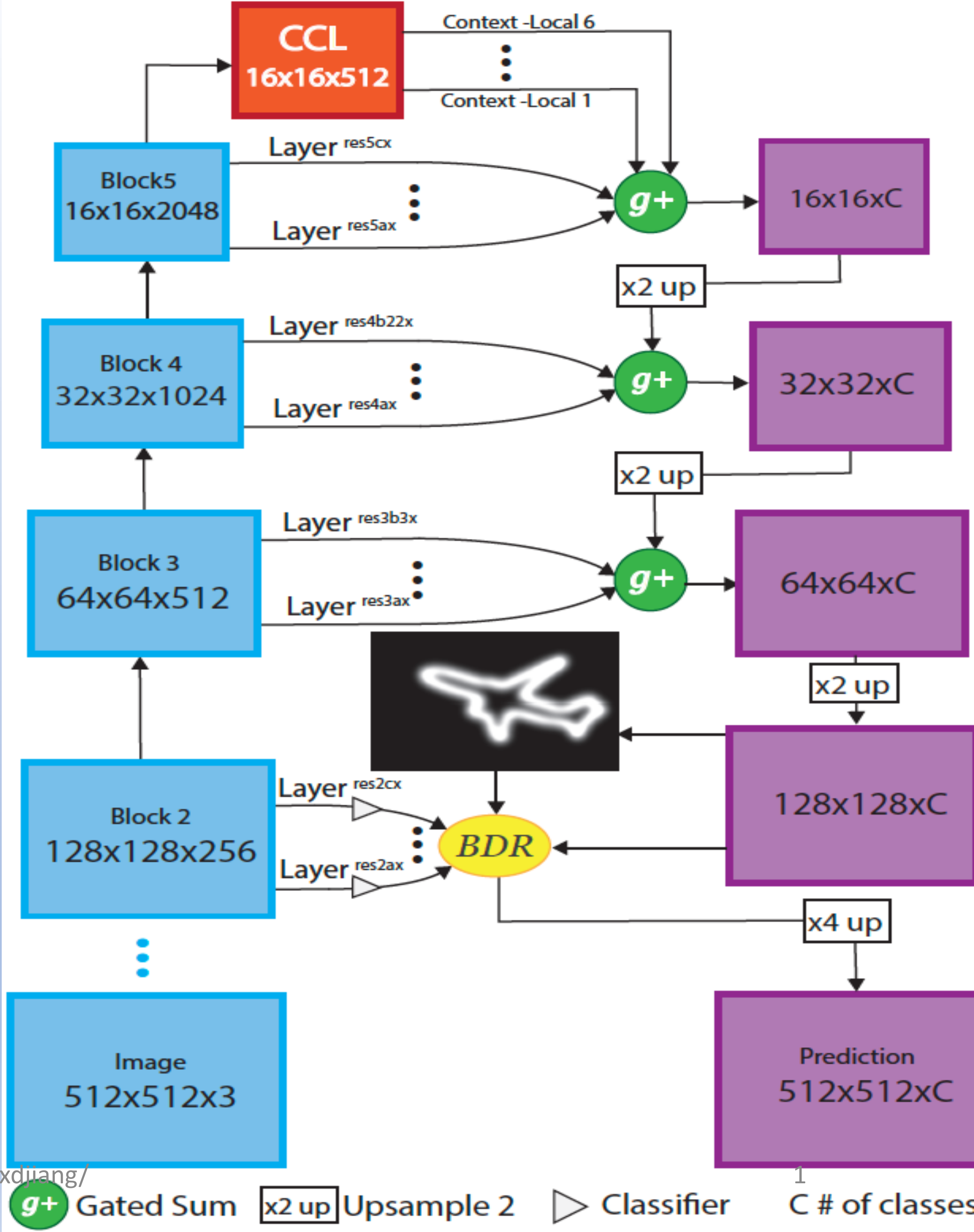
Further Development of CNN

Image Segmentation:

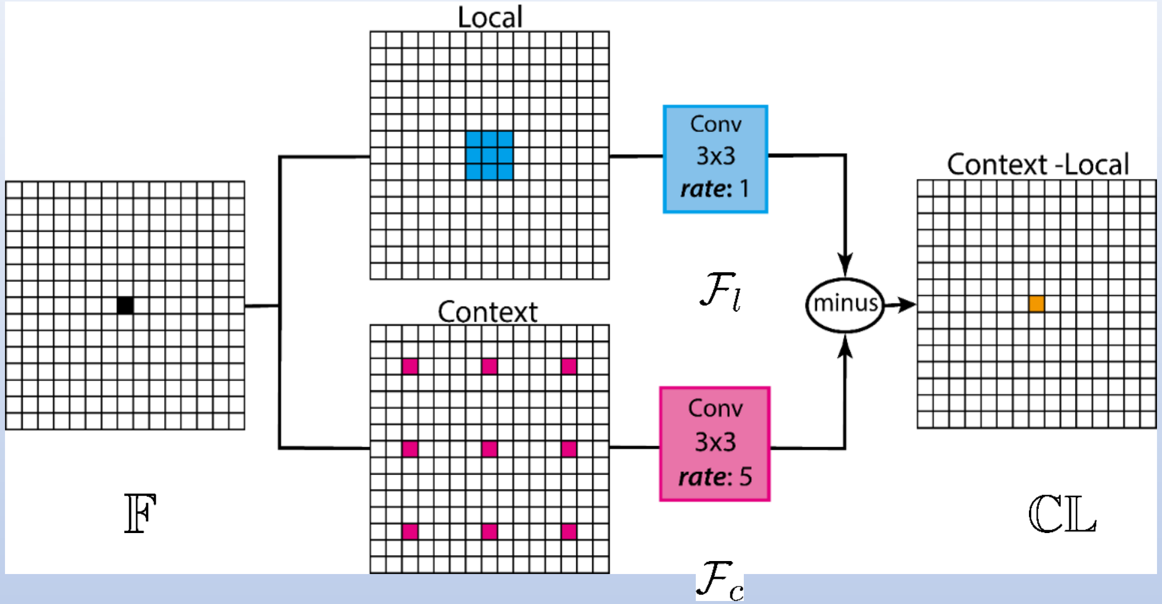
-- Pixel-wise Scene Understanding

H. Ding, X.D. Jiang, B. Shuai, A. Liu, and G. Wang, "Context contrasted feature and gated multi-scale aggregation for scene segmentation," ***CVPR Oral***, 2018.

H. Ding, X. Jiang, B. Shuai, A. Liu, G. Wang, "[Semantic Segmentation with Context Encoding and Multi-Path Decoding]," ***IEEE Transactions on Image Processing***, 2020.

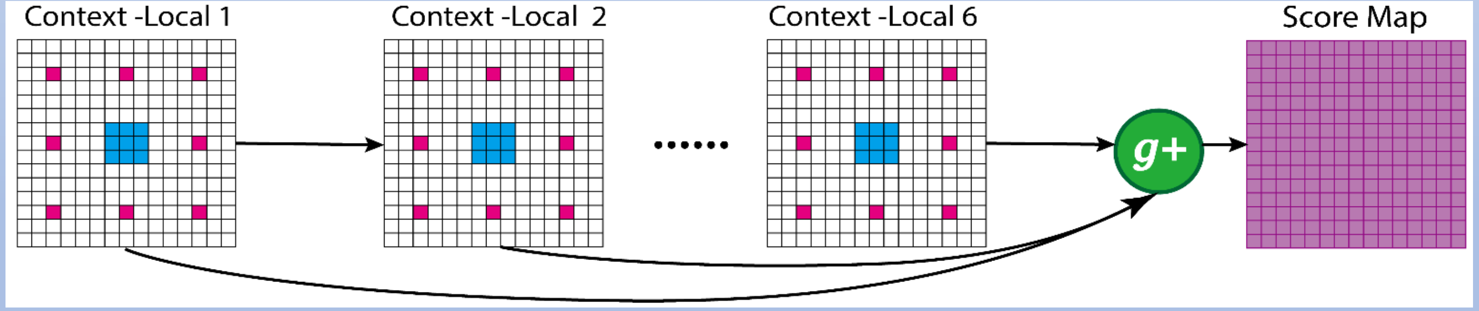


Context Contrasted Local Features



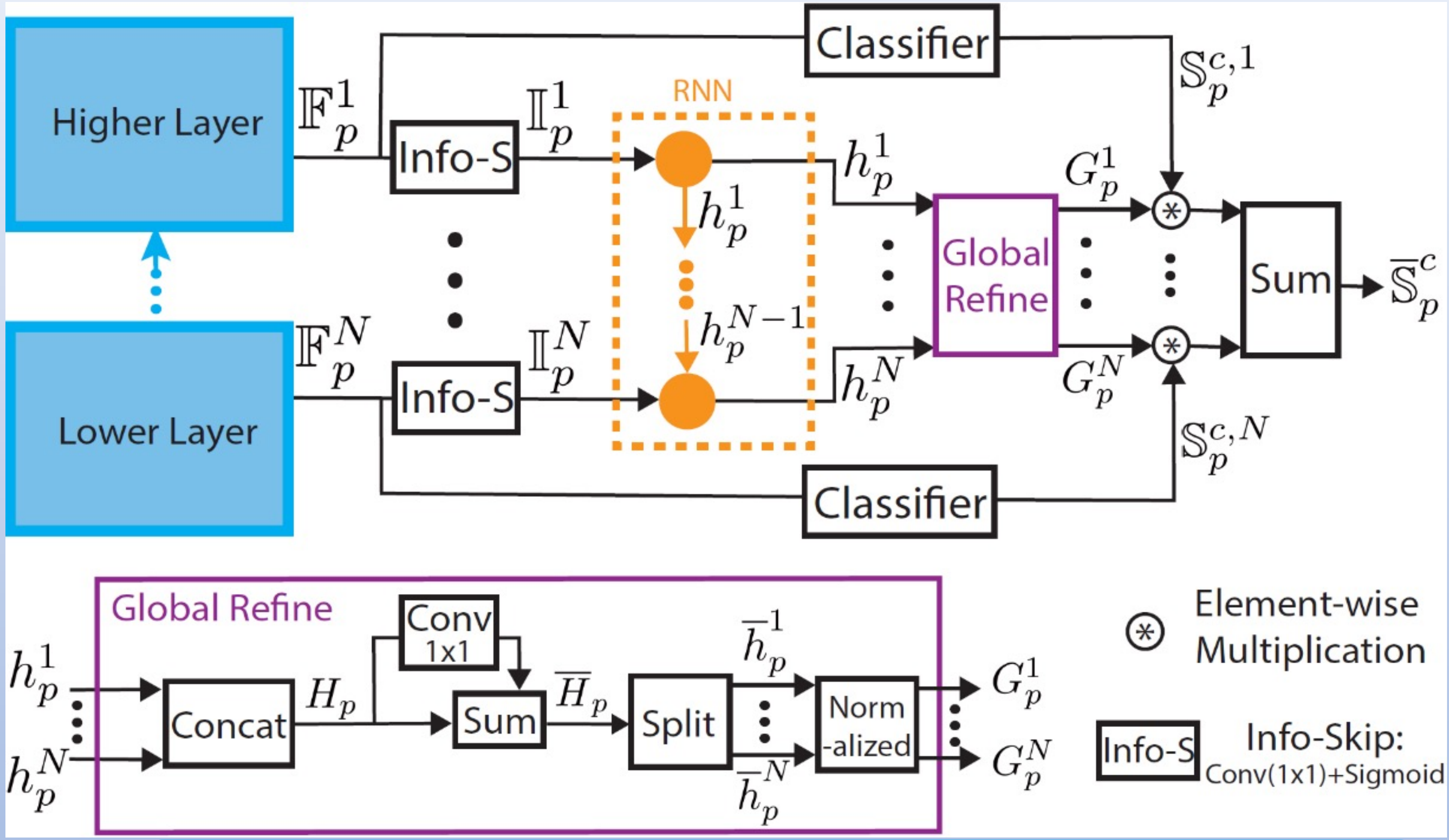
Context Contrasted Local Features:
The context contrasted local features are obtained via making a contrast between the local information and its context (surroundings).

$$\mathbb{CL} = \mathcal{F}_l(\mathbb{F}, \Theta_l) - \mathcal{F}_c(\mathbb{F}, \Theta_c)$$



Context Contrasted Local (CCL) Model:
Several blocks are chained to make multi-level context contrasted local features.

Gated Multi-scale Aggregation

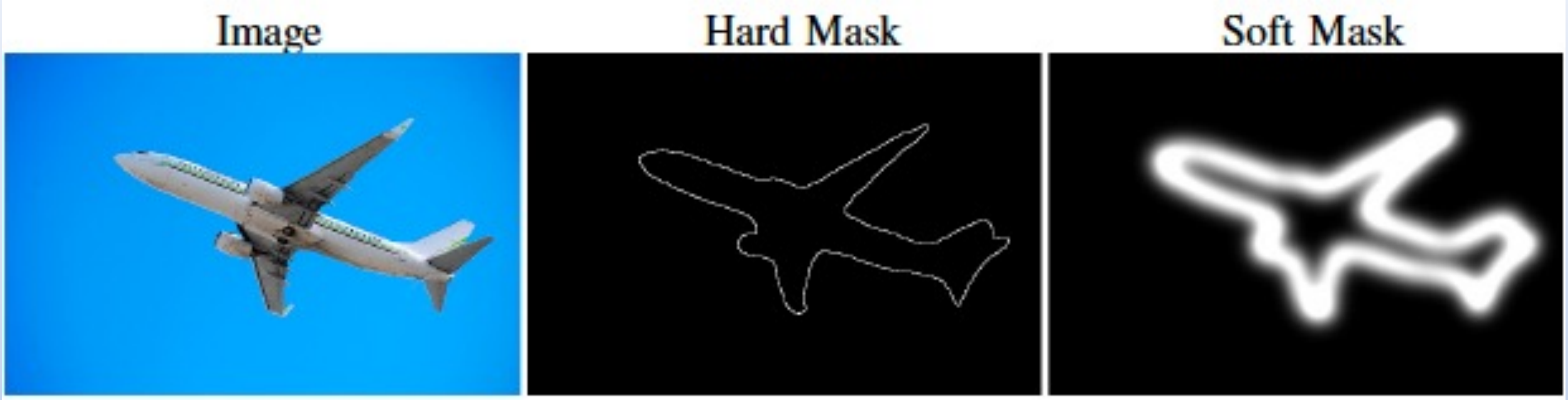


Gated Sum

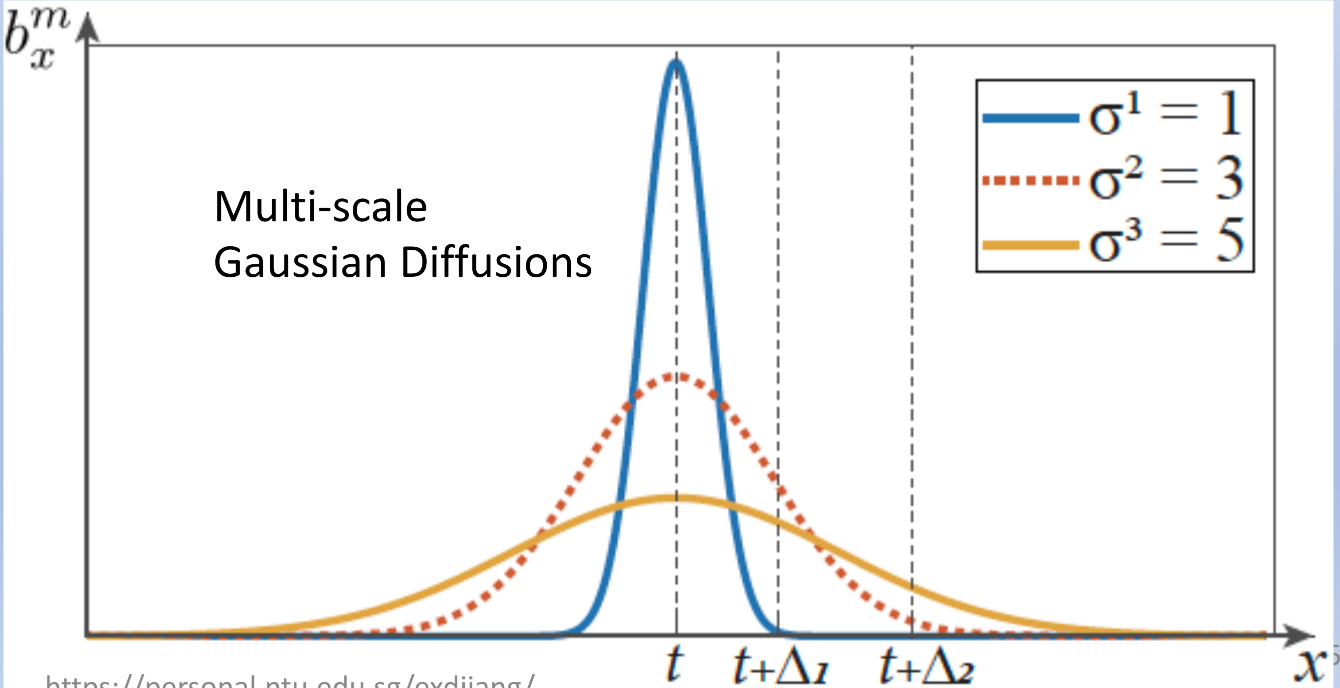
$$\overline{\mathbb{S}}_p^c = \sum_{n=1}^{N} G_p^n \mathbb{S}_p^{c,n}$$

The outputs of skip layers are selectively fused

Boundary Refinement

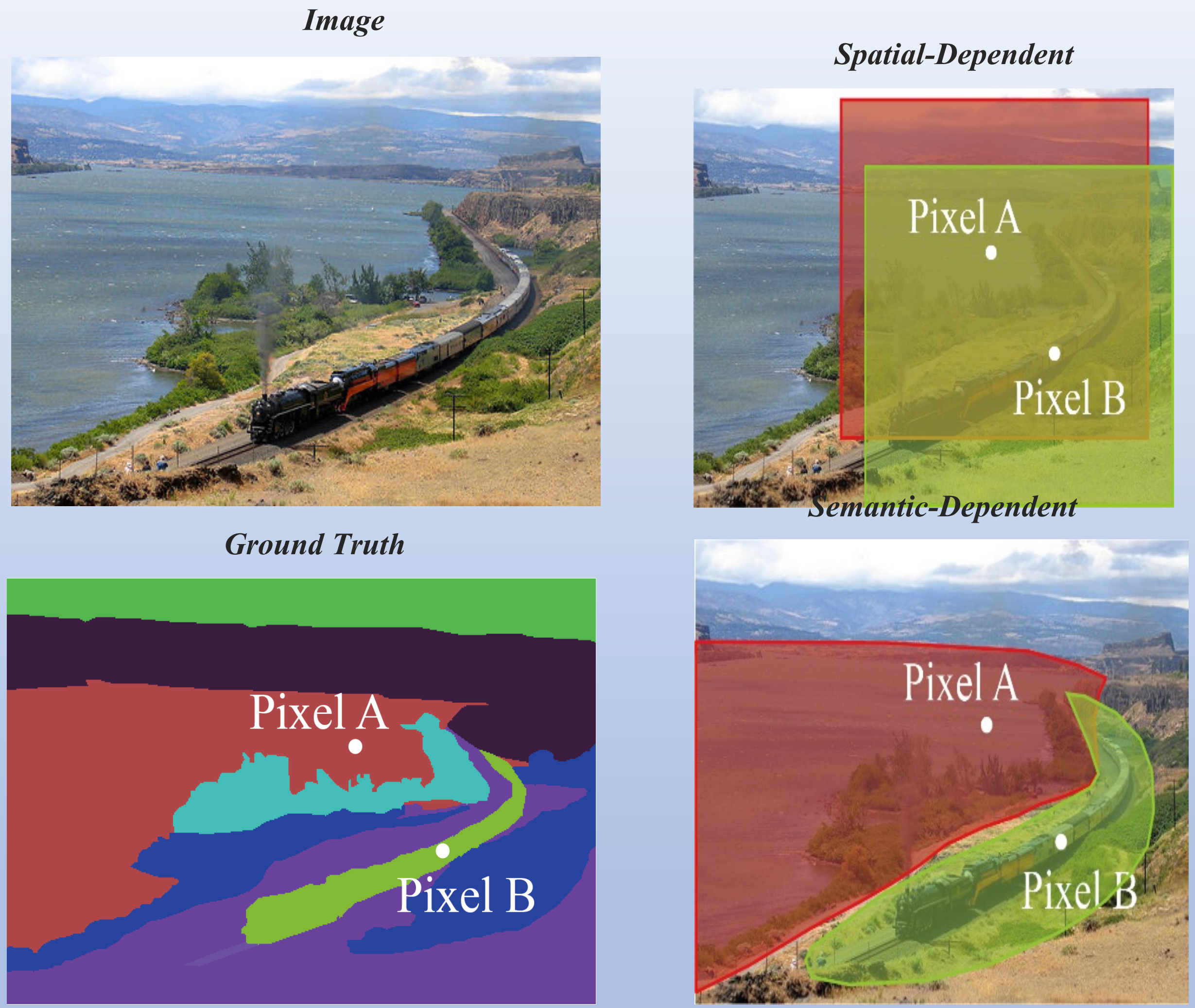


$$\widetilde{\mathbb{S}}_p^c = \sum_{m=1}^{M} \mathcal{B}_p^m \circledast \dot{\mathbb{S}}_p^{c,m} + \overline{\mathbb{S}}_p^c$$



Multi-scale
Gaussian Diffusions

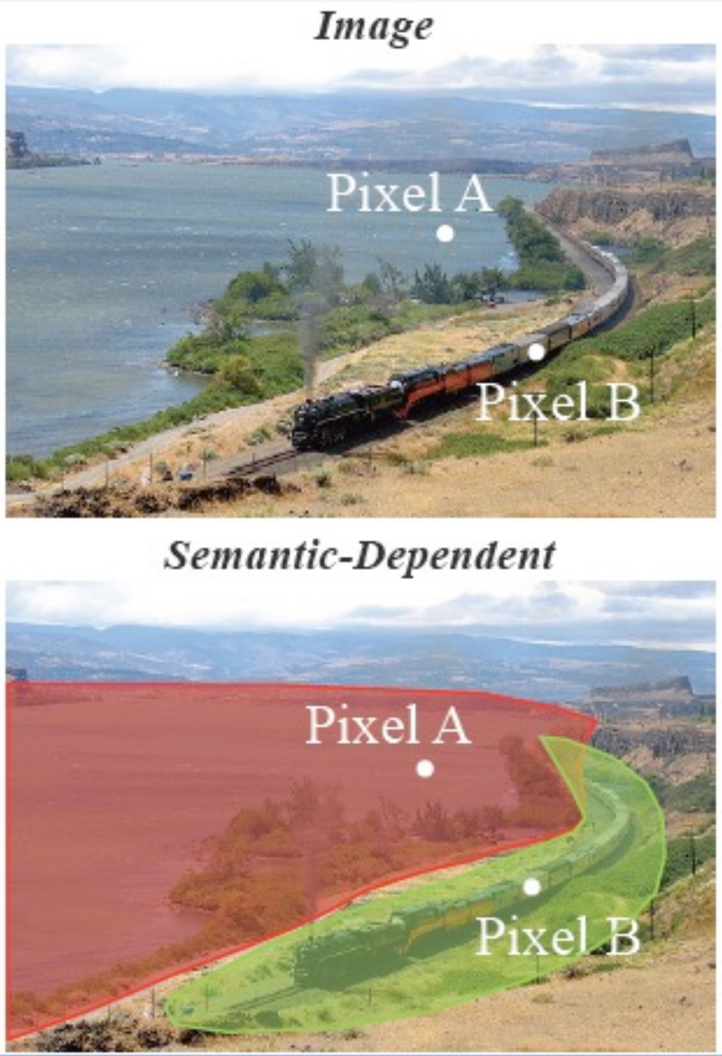
Shape Variant Convolution



Spatial-Dependent Context: with pre-defined windows (e.g., the red rectangle region for pixel A in the image)

Semantic-Dependent Context: with variant shapes according to the semantic correlation

H. Ding, X. Jiang, B. Shuai, A. Liu, G. Wang, "[Semantic Correlation Promoted Shape-Variant Context for Segmentation]," **CVPR'19 Oral**, 2019.



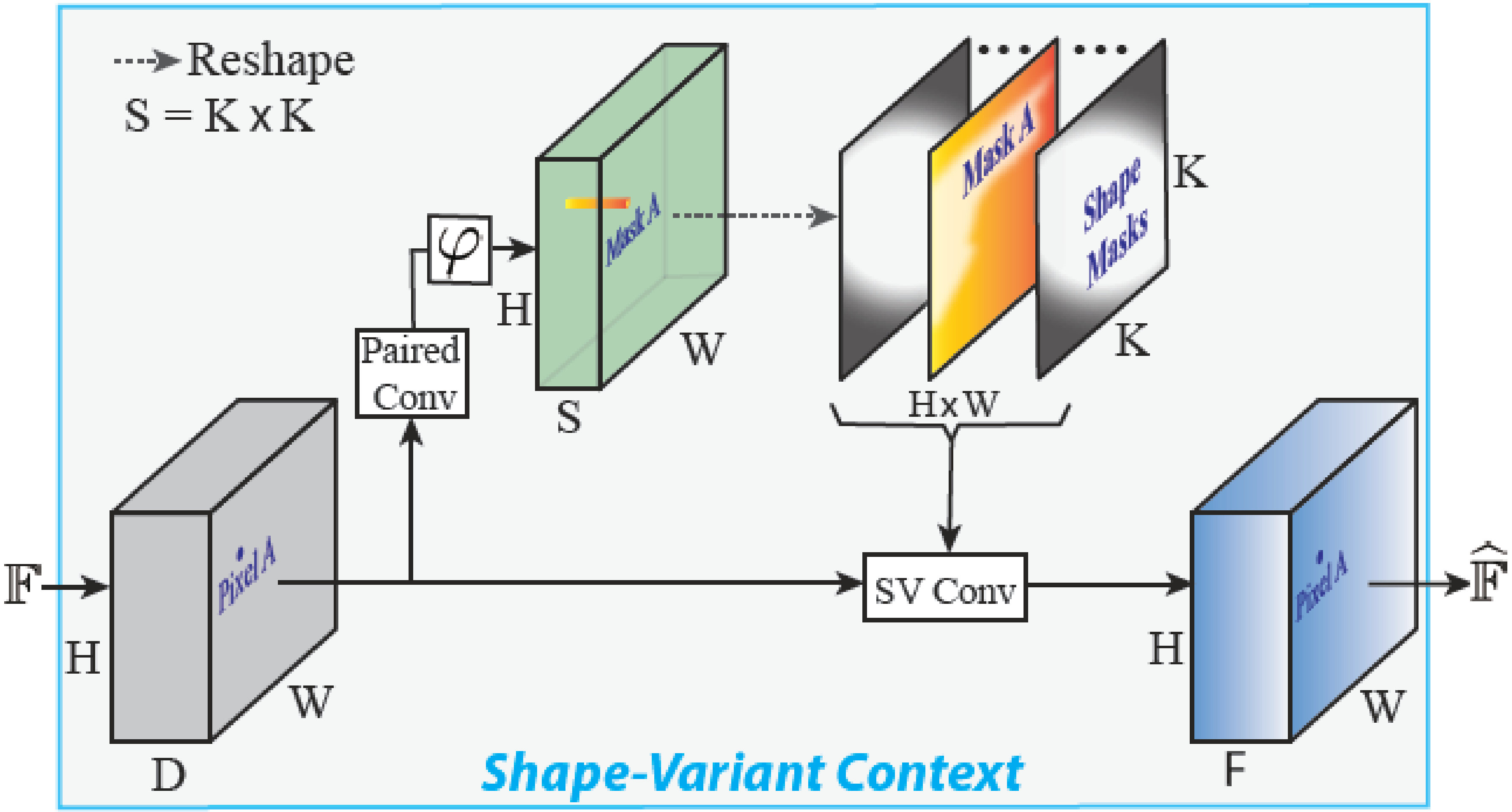
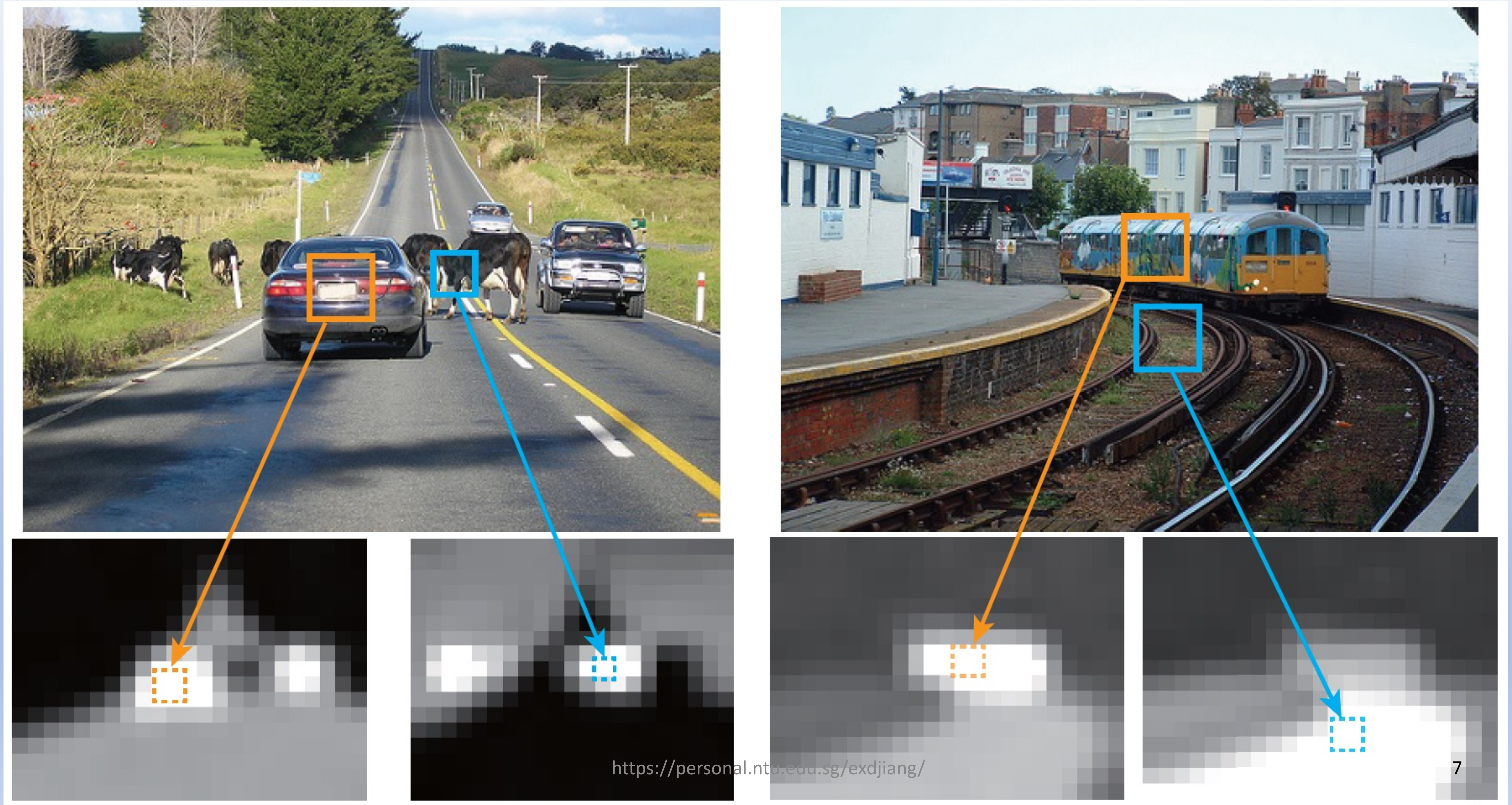


Figure 4. Semantic correlation-dependent shape-variant context aggregates surrounding information according to the semantic correlation and hence customizes an effective contextual region. It helps control the information flow within network via deciding what information to be passed or suppressed.

Four Visual Examples Shape Mask



Labeling De-noising in Decoding Process

Error of higher level : more on inaccurate location

Error of lower level : more on noisy classes

$$\mathcal{E}_k = F_g(F_{sf}(\mathcal{S}_k))$$

\# training classes: 500

\# classes in an input image: 5

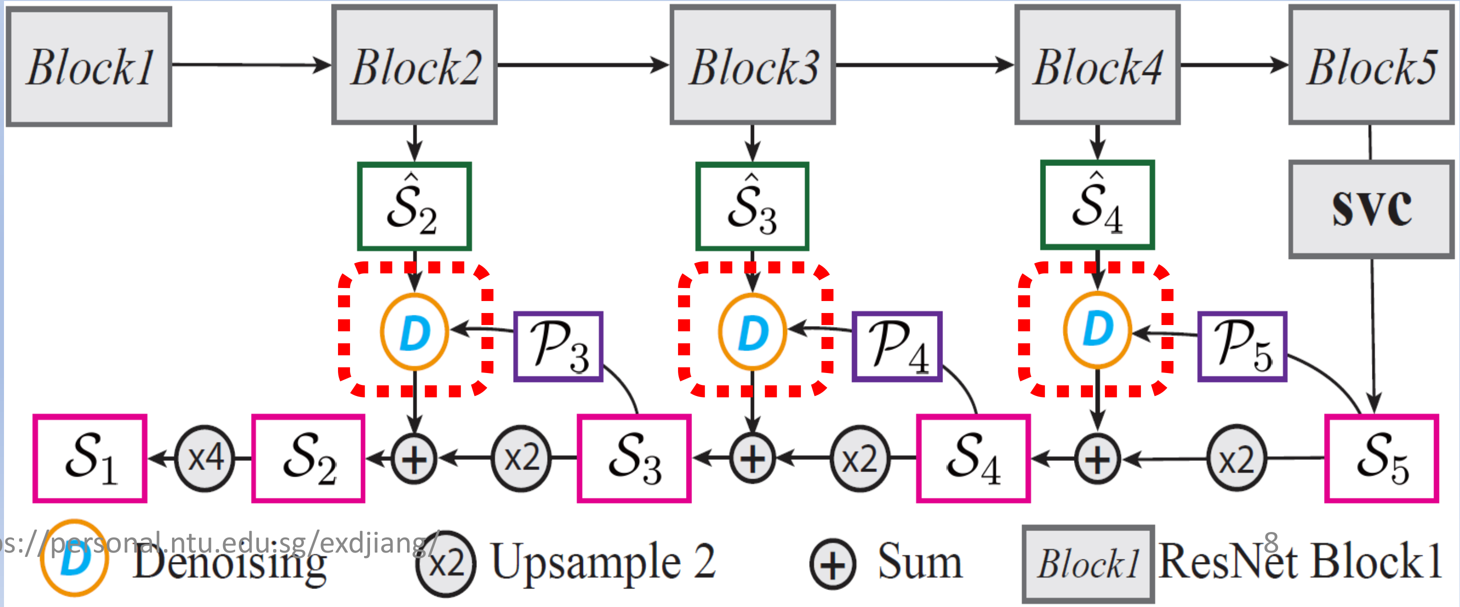
\# of classified classes of layers

90th, 70th, 50th, 30th

5, 8, 14, 22

$$\mathcal{P}_k^c = \mathrm{ReLU}(\mathcal{T} - e_k^c)\Delta_k^c$$

$$\mathcal{S}_{k-1}^c = \mathrm{ReLU}(\hat{\mathcal{S}}_{k-1}^c - \mathcal{P}_k^c) + \mathcal{S}_k^c$$



Understand Transformer

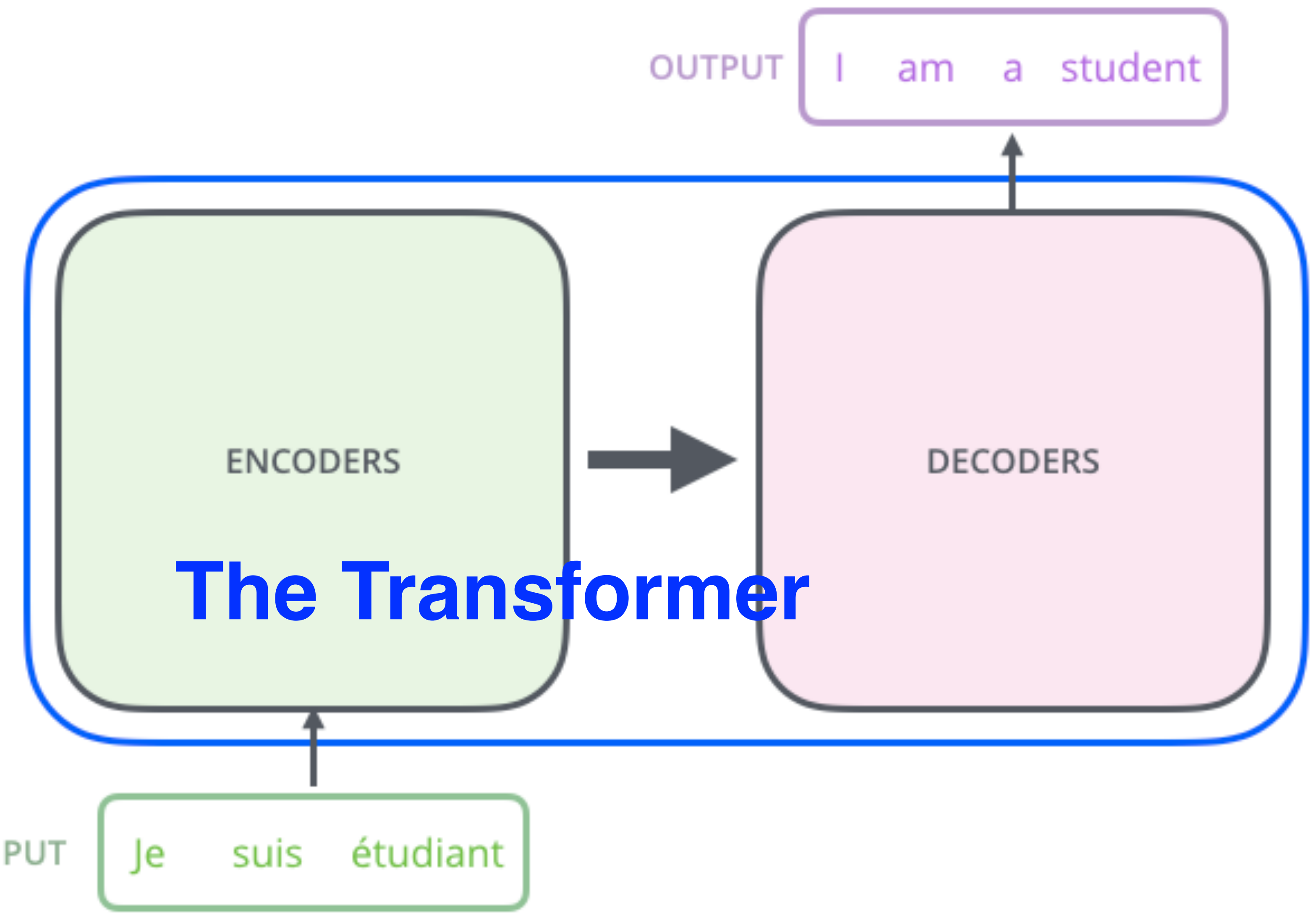
- The Transformer has a neural network architecture that transforms an input sequence to an output sequence such as speech, text and time series.
- Recurrent neural networks (RNNs) and its further development, Long-Short Term Memory (LSTM), are clearly related to sequences and lists. Transformer outperforms them by parallelization of their sequential operations.
- It is developed originally for natural language processing (NLP). Now it becomes a powerful neural network architecture also for computer vision, showing more powerful than CNN.
- The Transformer gets its powers thanks to its Attention module. It captures the relationships between each word/token in a sequence with every other word/token. The original paper of transformer: ”Attention Is All You Need”.
- how does the transformer work? Is attention really everything? Any relation to CNN?

Transformer consists of Encoders and Decoders

A machine translation application, it takes a sentence in one language, and outputs its translation in another.

All words/tokens of a sentence/image are parallelly inputted into the transformer.

Output is a linear classification of the decoder output.

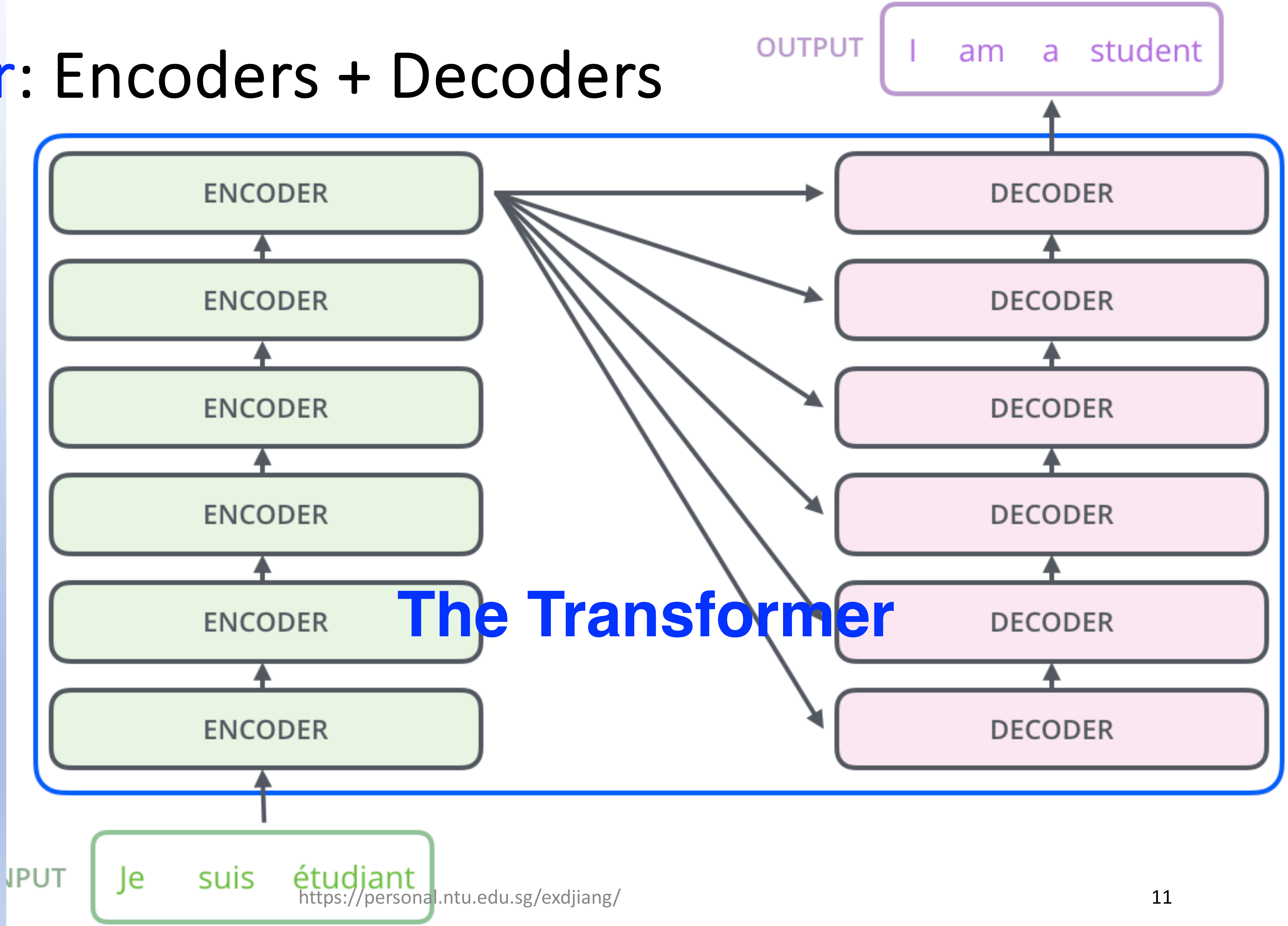


Transformer: Encoders + Decoders

A stack of encoders of the same structure

and

a stack of decoders of the same structure

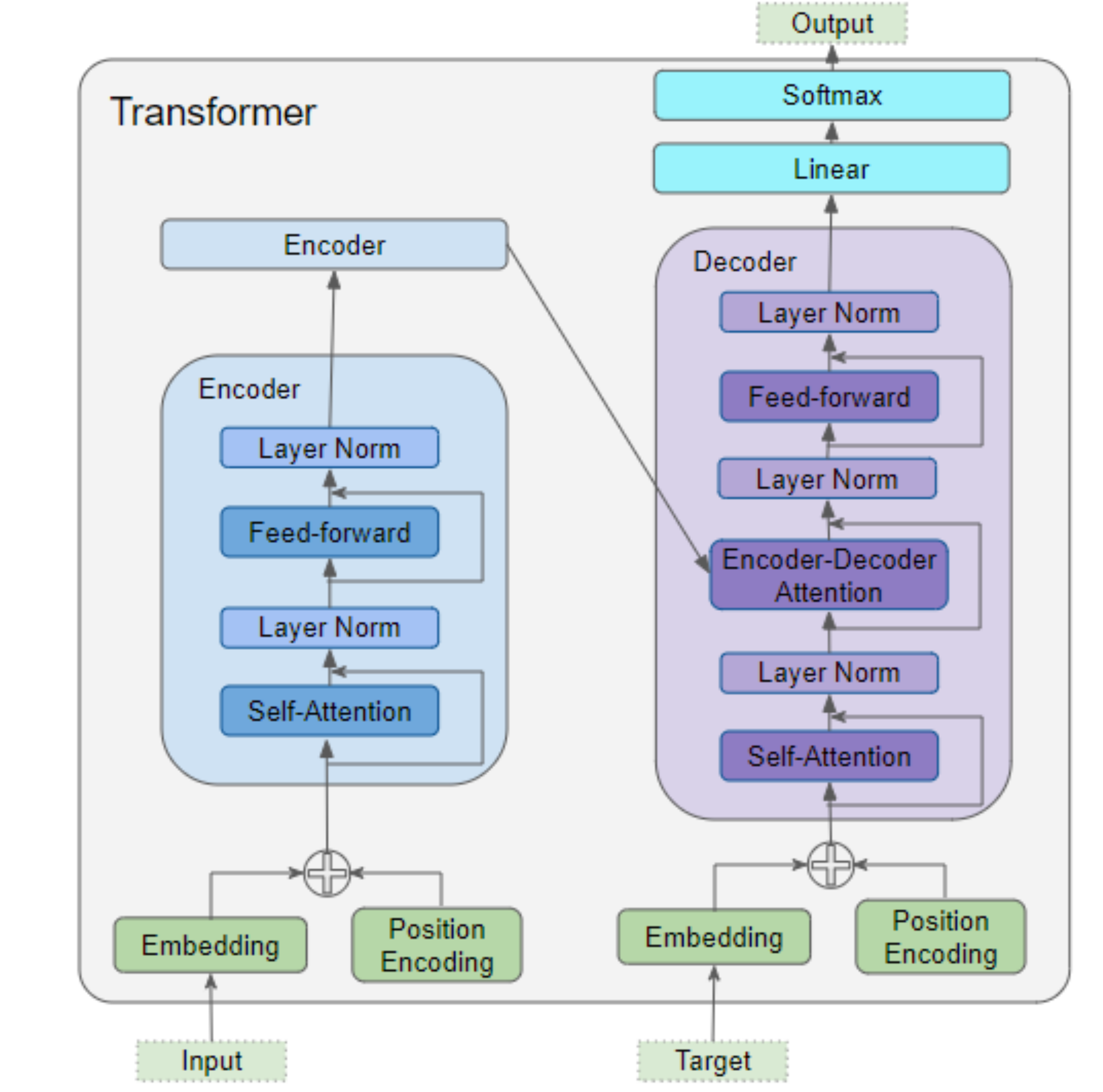


Transformer:
Encoders + Decoders

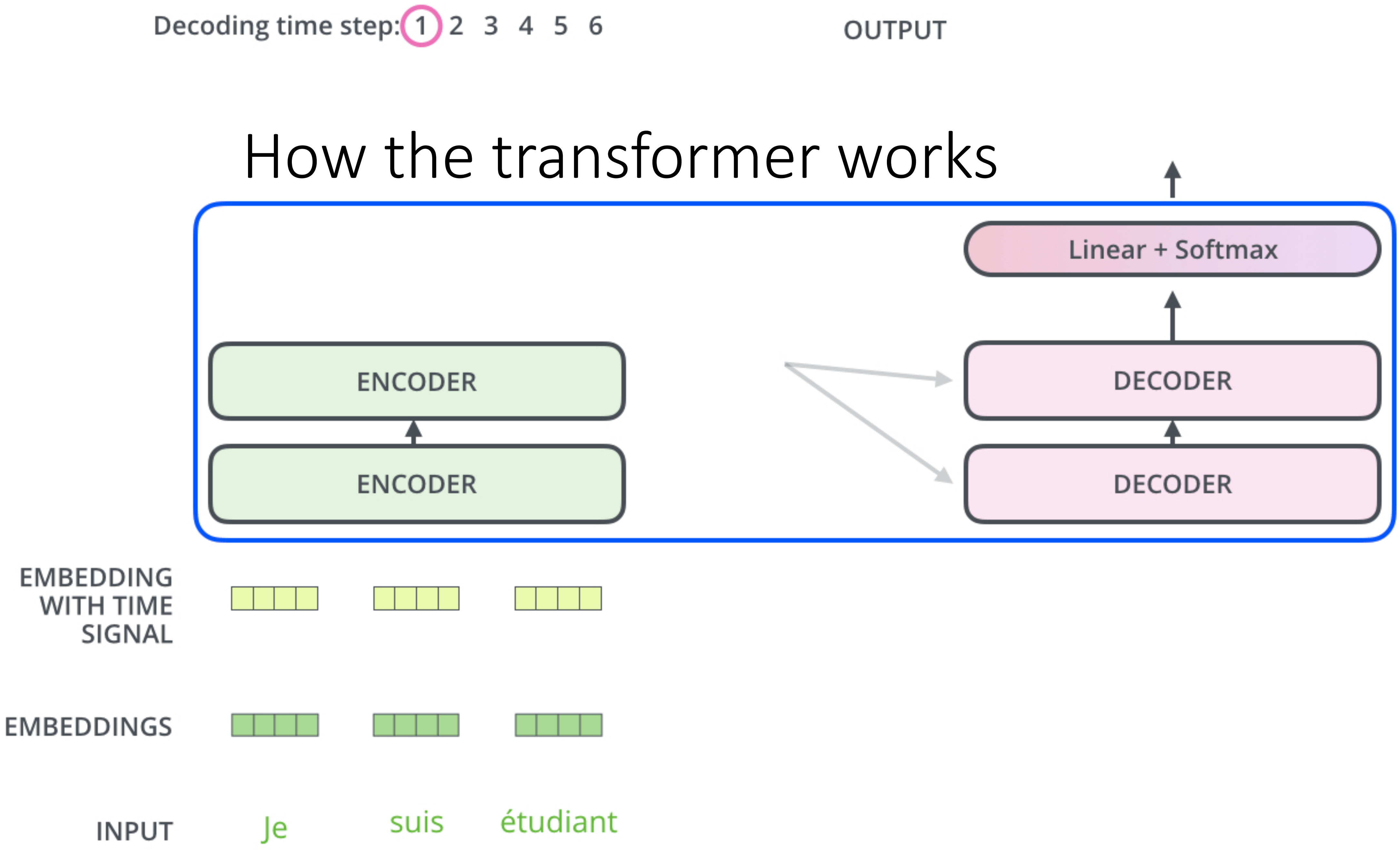
Structures of an Encoder and a Decoder

Commonality and Difference of encoder and decoder

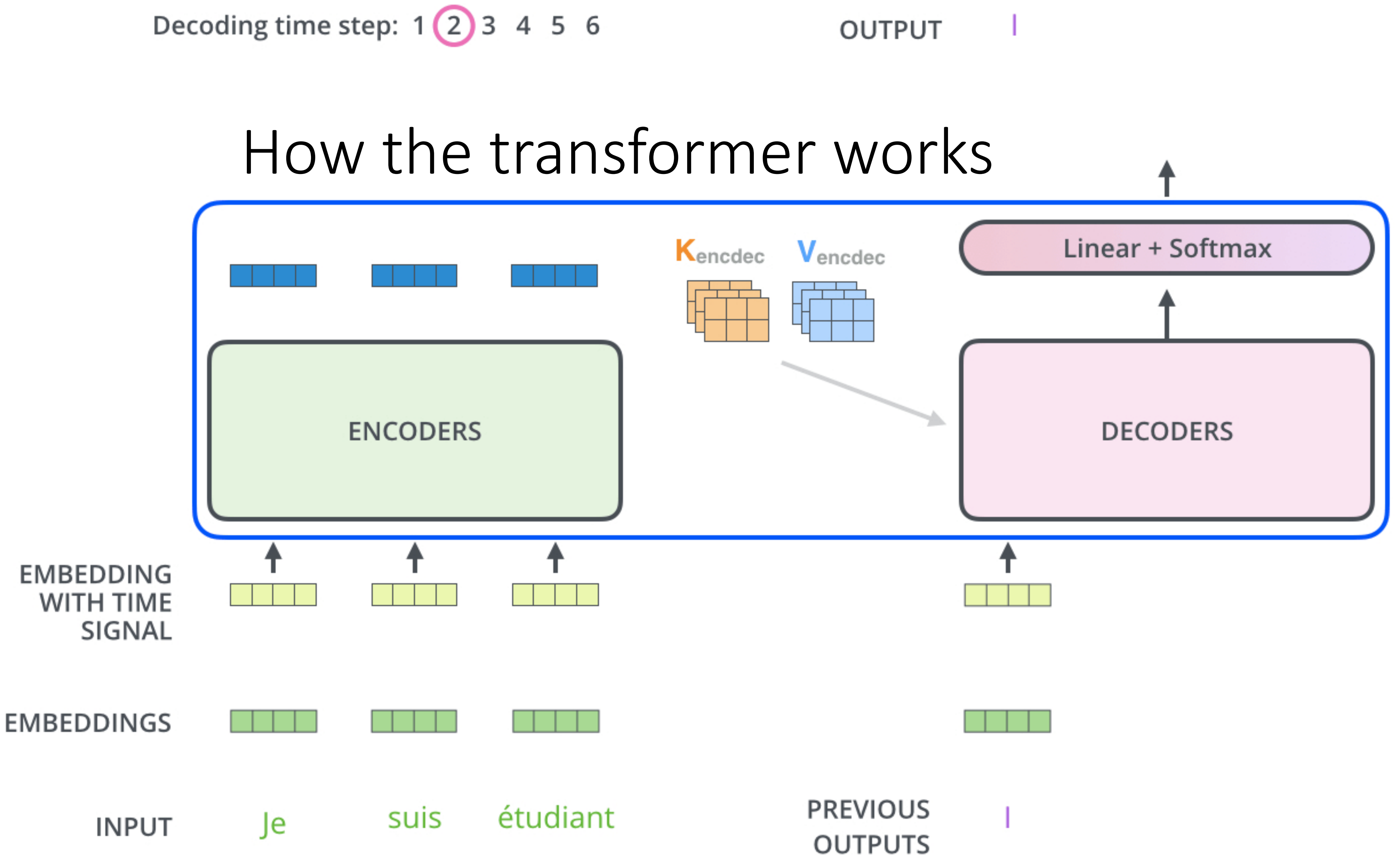
https:/



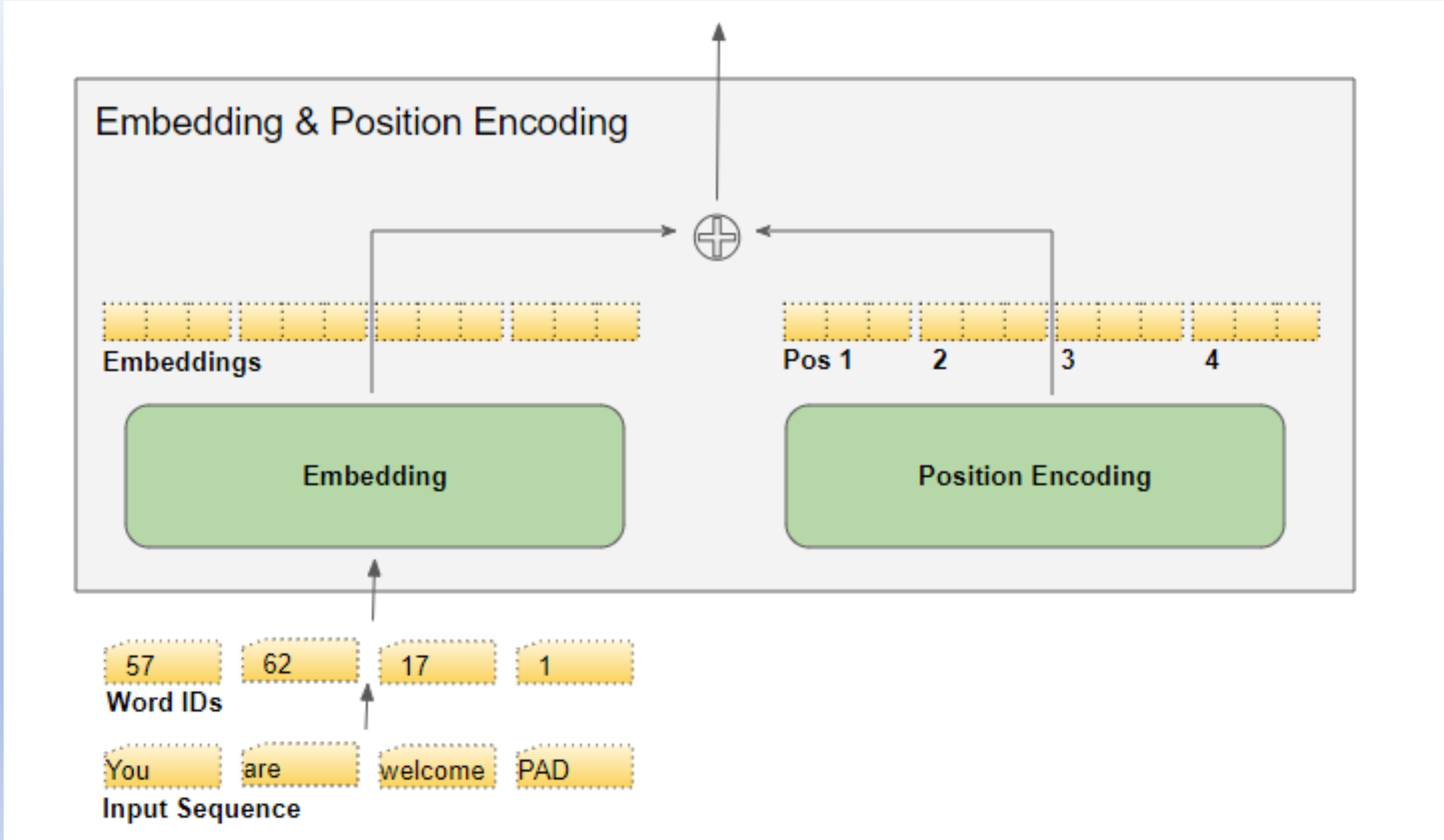
How the transformer works



How the transformer works



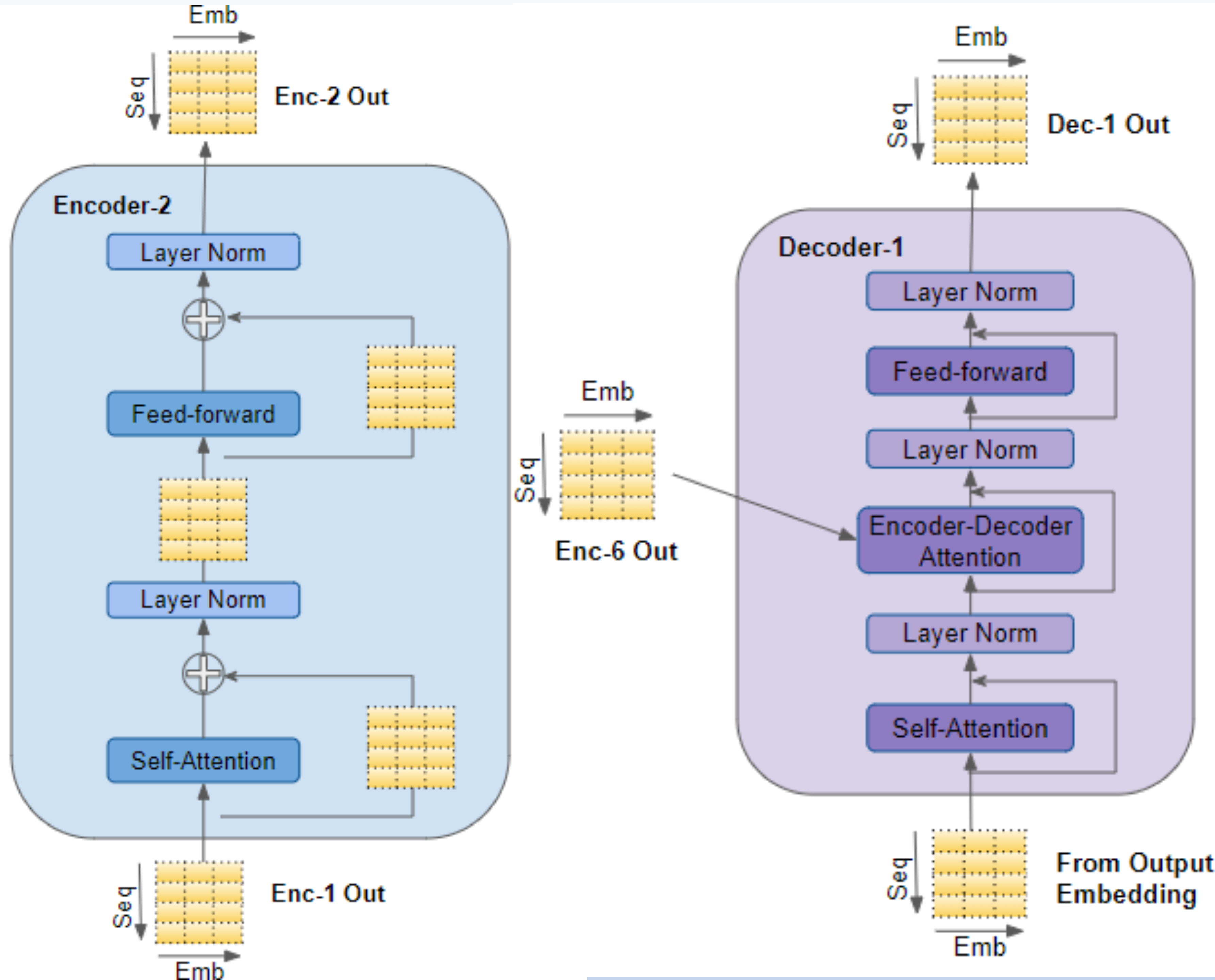
Embedding each input word/token into a feature vector



Both the Attention and Feed-forward layers, have a residual skip-connection.

The pointwise feed-forward network is a couple of linear layers with a ReLU activation in between.

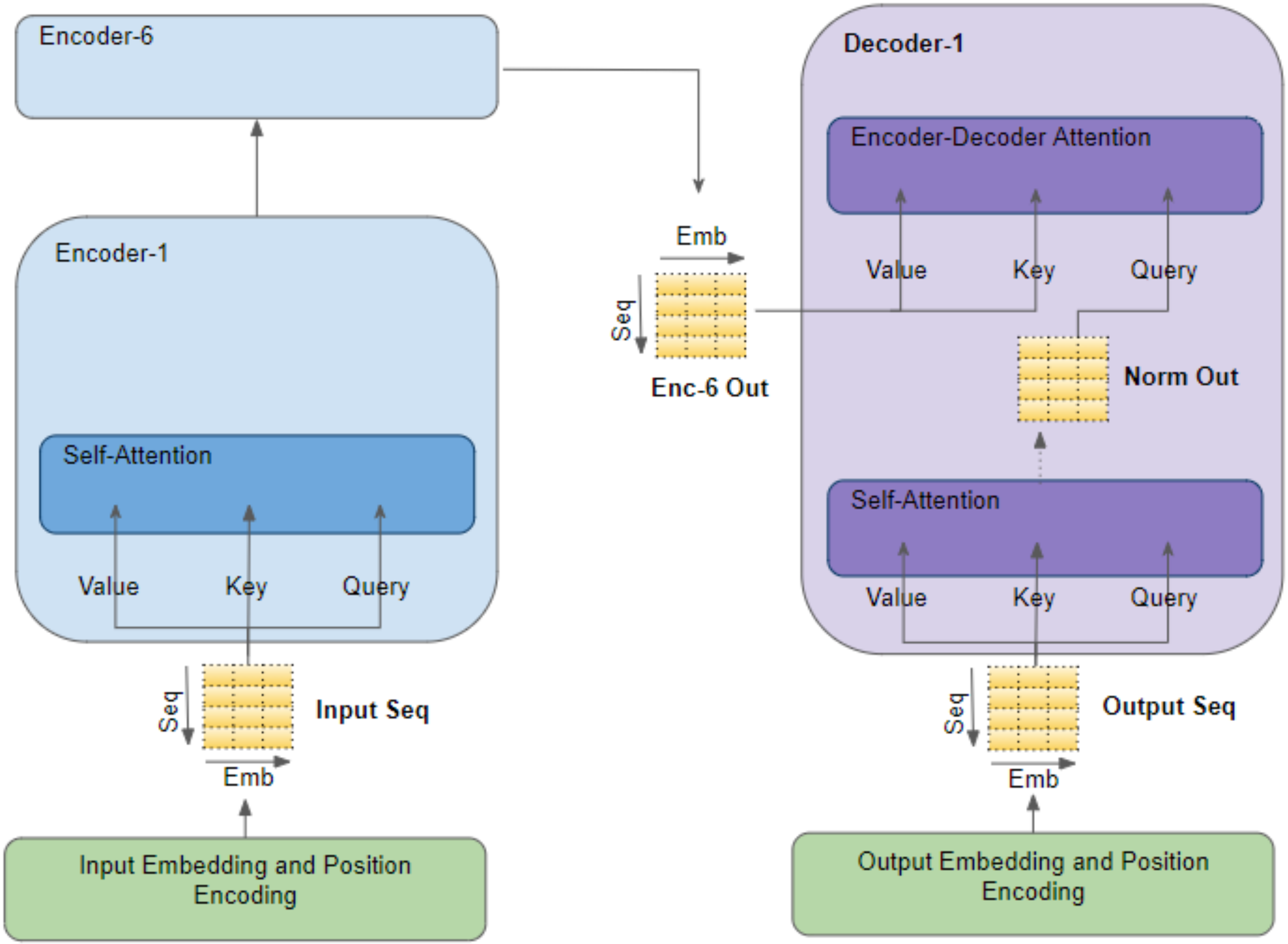
Pointwise?



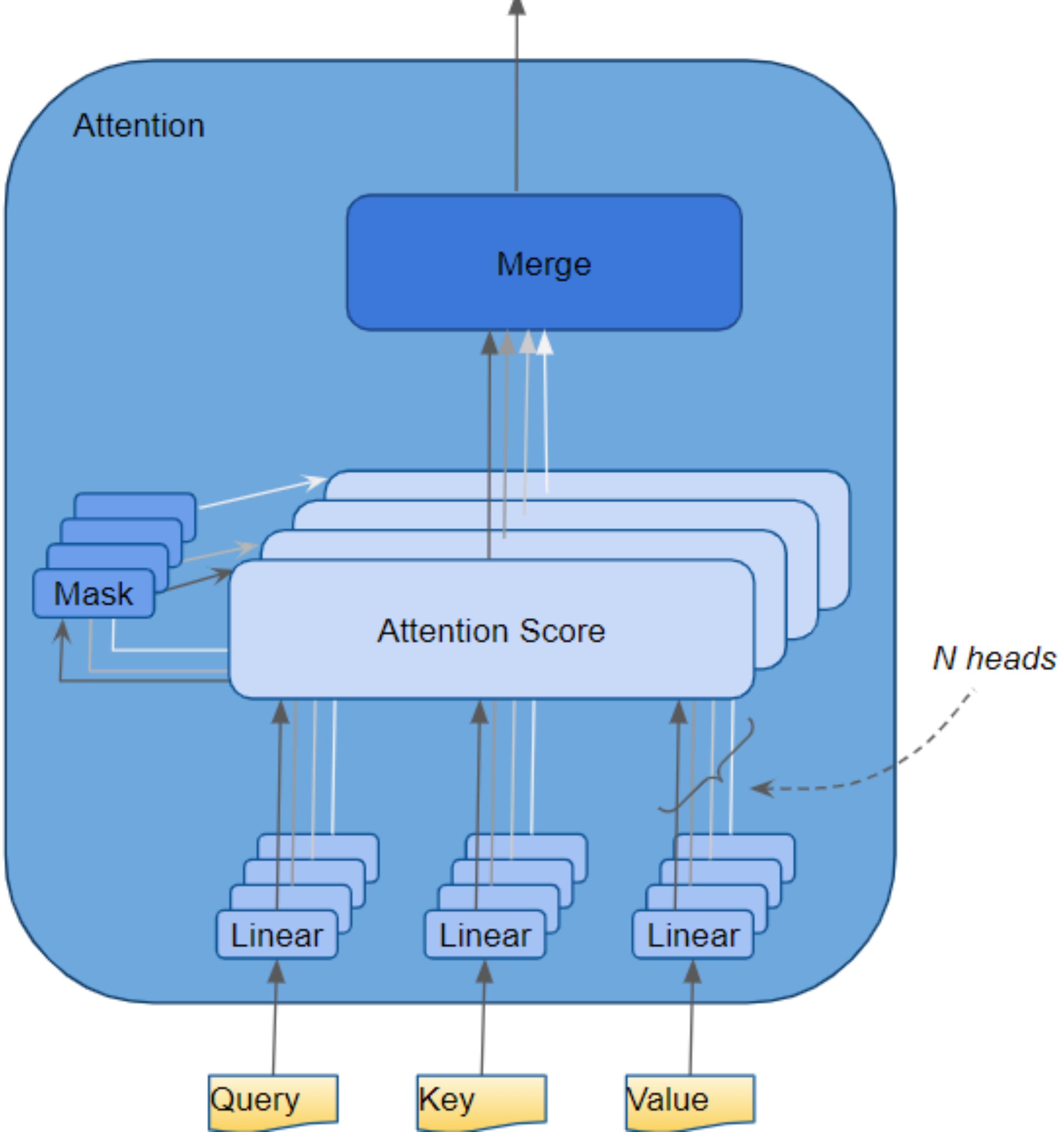
Self-attention

and

Encoder-decoder attention



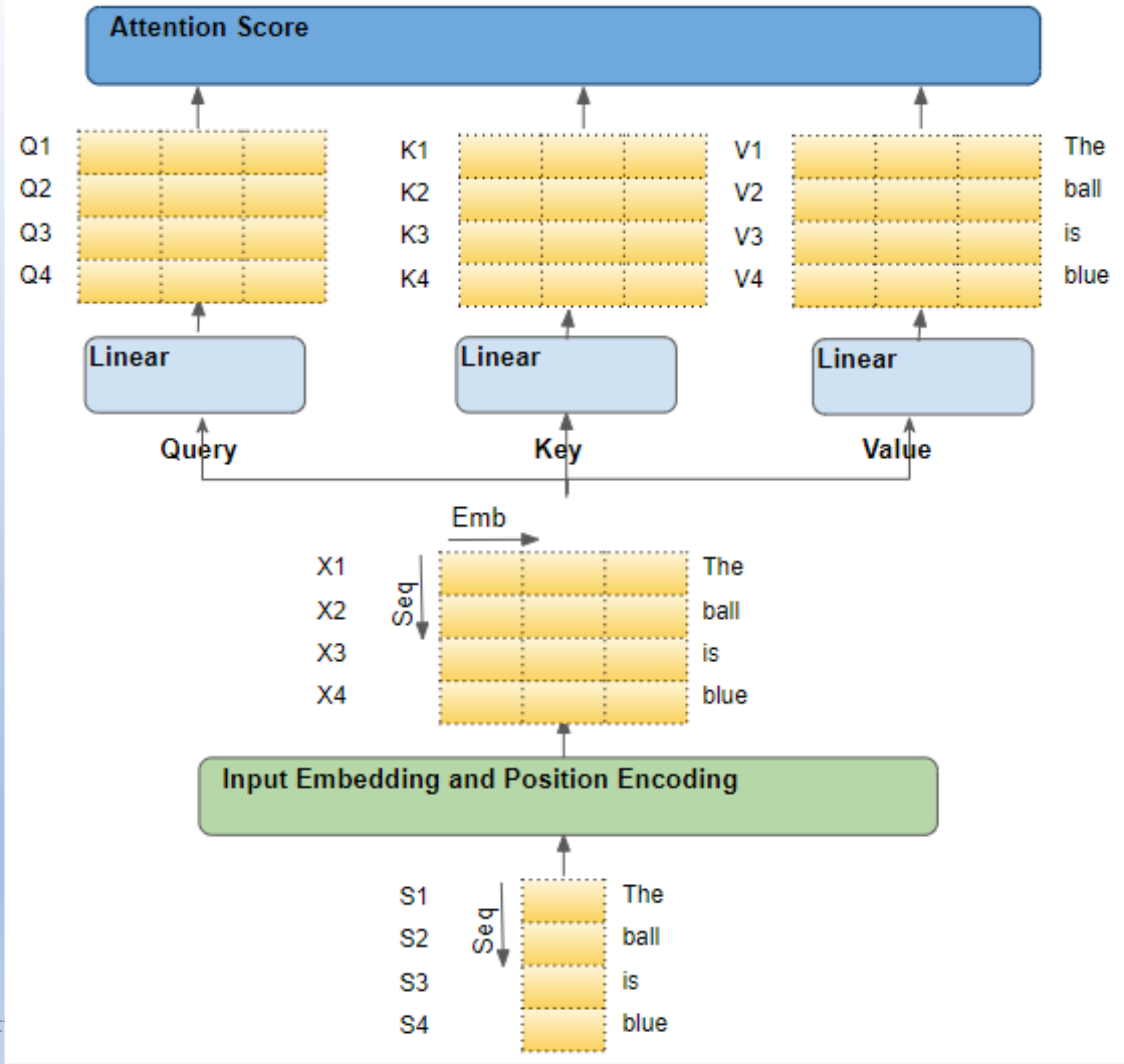
Multiple attention heads in each encoder and decoder.



Prepare for Attention

An example of English-to-Spanish translation problem, where one sample source sequence is “The ball is blue”. The target sequence is “La bola es azul”.

Three copies of each word/token are generated for self-attention by linear projection.



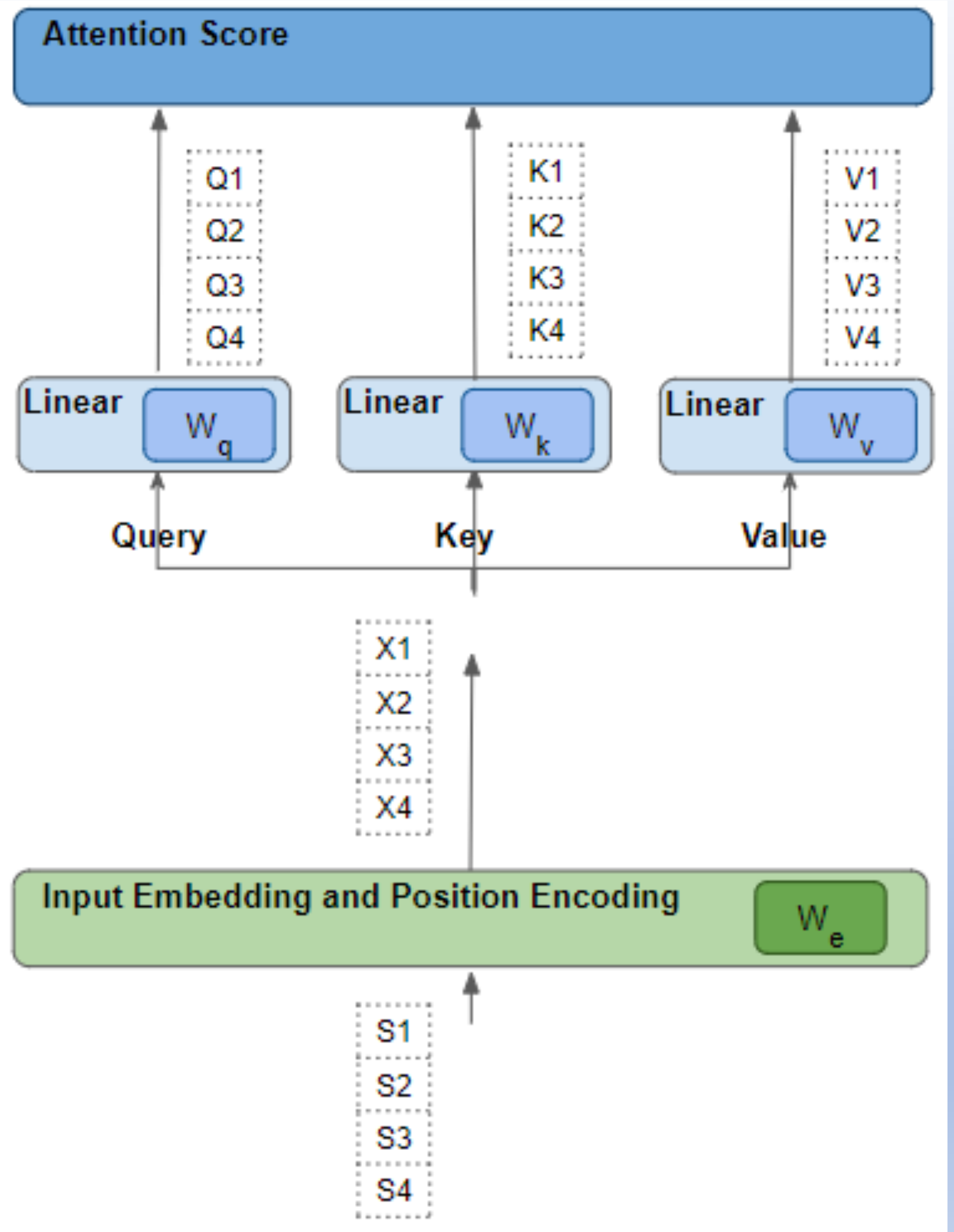
Learnable linear Projections:

The input sequence (a matrix) is passed through three trainable linear layers which produce three separate matrices — known as the Query, Key, and Value.

Let $\mathbf{X} = \begin{pmatrix} X1 \\ X2 \\ X3 \\ X4 \end{pmatrix}$, $\mathbf{Q} = \begin{pmatrix} Q1 \\ Q2 \\ Q3 \\ Q4 \end{pmatrix}$, $\mathbf{K} = \begin{pmatrix} \mathrm{K1} \\ \mathrm{K2} \\ \mathrm{K3} \\ \mathrm{K4} \end{pmatrix}$, $\mathbf{V} = \begin{pmatrix} \mathrm{V1} \\ \mathrm{V2} \\ \mathrm{V3} \\ \mathrm{V4} \end{pmatrix}$

Then: $\mathbf{Q} = \mathbf{X}W_q$, $\mathbf{K} = \mathbf{X}W_k$, $\mathbf{V} = \mathbf{X}W_v$

The important thing to keep in mind is that each ‘row’ of these matrices corresponds to one word (token) in the source sequence.



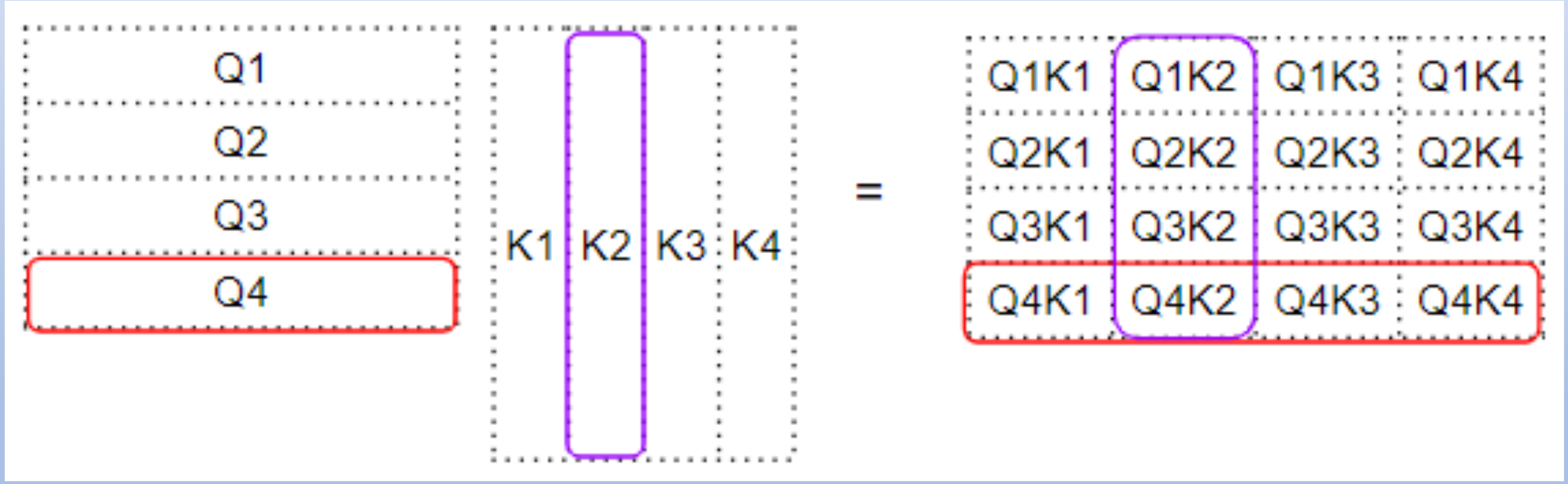
Attention Score — Dot Product between Q and K words

- The first step of Attention is to do a matrix multiply (ie. dot product) between the Query (Q) matrix and a transpose of the Key (K) matrix.

$$R = QK^T$$

- Watch what happens to each word. **Dot product generates similarity between words**

$$R = QK^T$$



- We produce an intermediate matrix (let's call it a 'factor' matrix) where each cell is a matrix multiplication between two words. **Covariance Matrix!!!**

Attention: Attended (weighted) combination of Values

The dot product between the Query and Key computes the relevance between each pair of words. This relevance is then used as a “factor” to compute a weighted sum of all the Value words. That weighted sum is output as the Attention module.

$$\begin{bmatrix} Q1K1 & Q1K2 & Q1K3 & Q1K4 \\ Q2K1 & Q2K2 & Q2K3 & Q2K4 \\ Q3K1 & Q3K2 & Q3K3 & Q3K4 \\ \boxed{Q4K1 \quad Q4K2 \quad Q4K3 \quad Q4K4} \end{bmatrix} \begin{bmatrix} V1 \\ \boxed{V2} \\ V3 \\ V4 \end{bmatrix} = \begin{bmatrix} Q1K1V1 + Q1K2V2 + Q1K3V3 + Q1K4V4 \\ Q2K1V1 + Q2K2V2 + Q2K3V3 + Q2K4V4 \\ Q3K1V1 + Q3K2V2 + Q3K3V3 + Q3K4V4 \\ \boxed{Q4K1V1 + Q4K2V2 + Q4K3V3 + Q4K4V4} \end{bmatrix}$$

$$Z = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V = \begin{bmatrix} Z1 \\ Z2 \\ Z3 \\ \boxed{Z4} \end{bmatrix}$$

Fourth word Score

Fourth Query word * first Key word

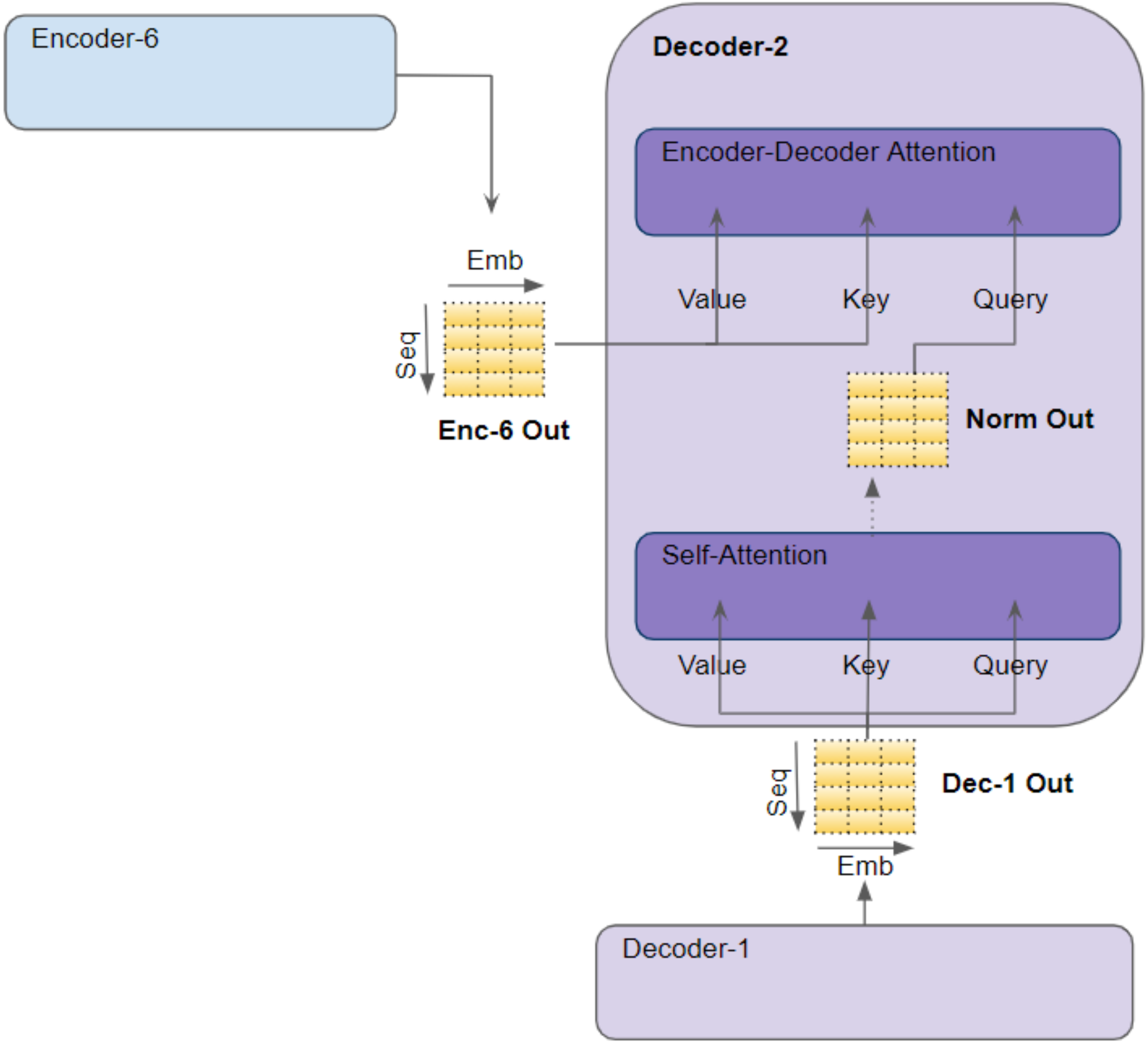
$$Z_4 = (Q_4K_1)\,V_1 + (Q_4K_2)\,V_2 + (Q_4K_3)\,V_3 + (Q_4K_4)\,V_4$$

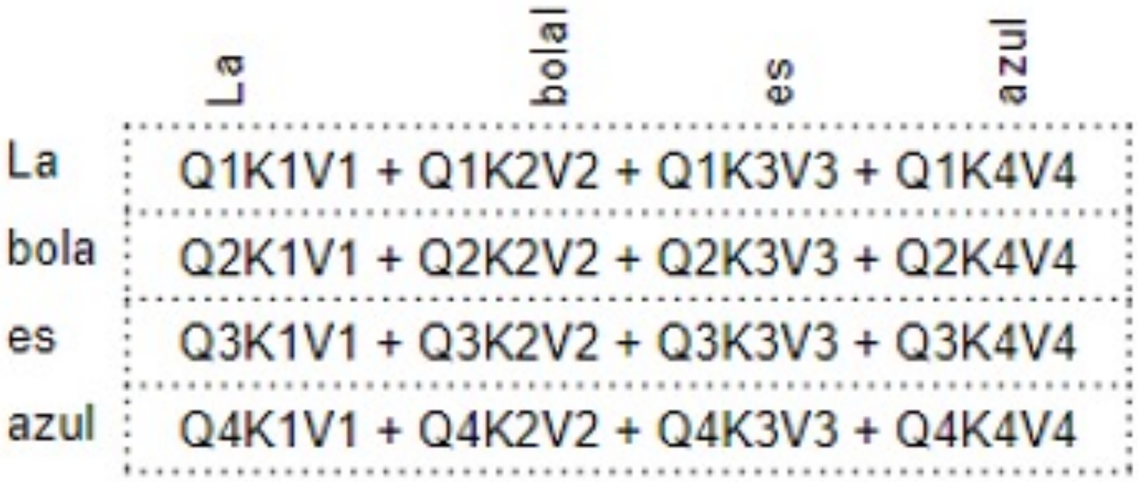
Self-attention in Encoder — the source sequence pays attention to itself.

Fourth Query word * second Key word

Encoder-Decoder attention

In the Decoder's Encoder-Decoder attention, the output of the final Encoder in the stack is passed to the Value and Key parameters. The output of the Self-attention module below it is passed to the Query parameter.





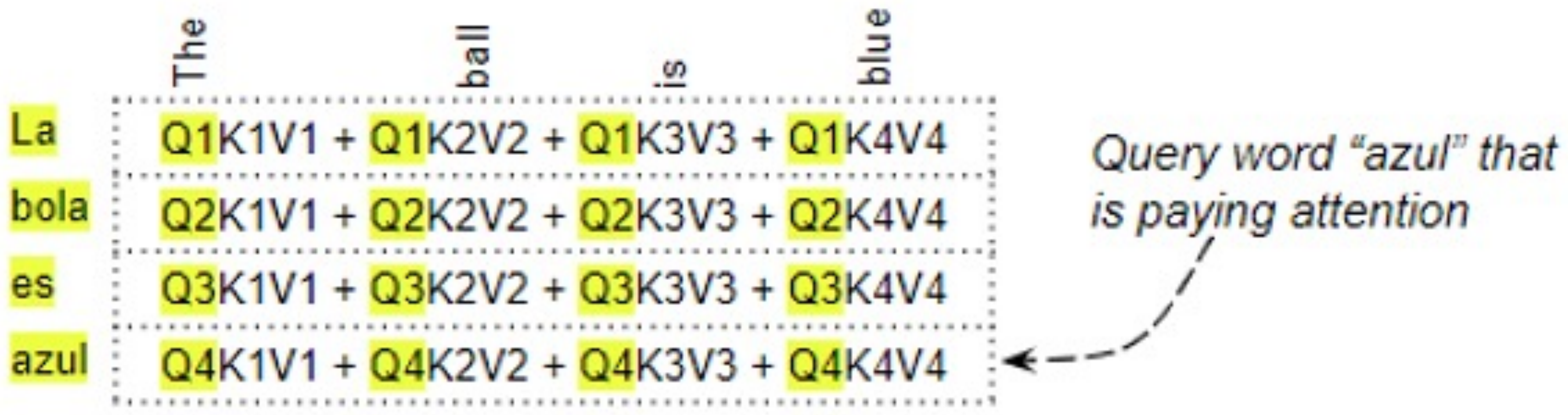
Decoder Self Attention

Target sentence paying attention to itself

Self-attention in the Decoder — the target sequence pays attention to itself.

Encoder-Decoder-attention in the Decoder — the target sequence pays attention to the source sequence

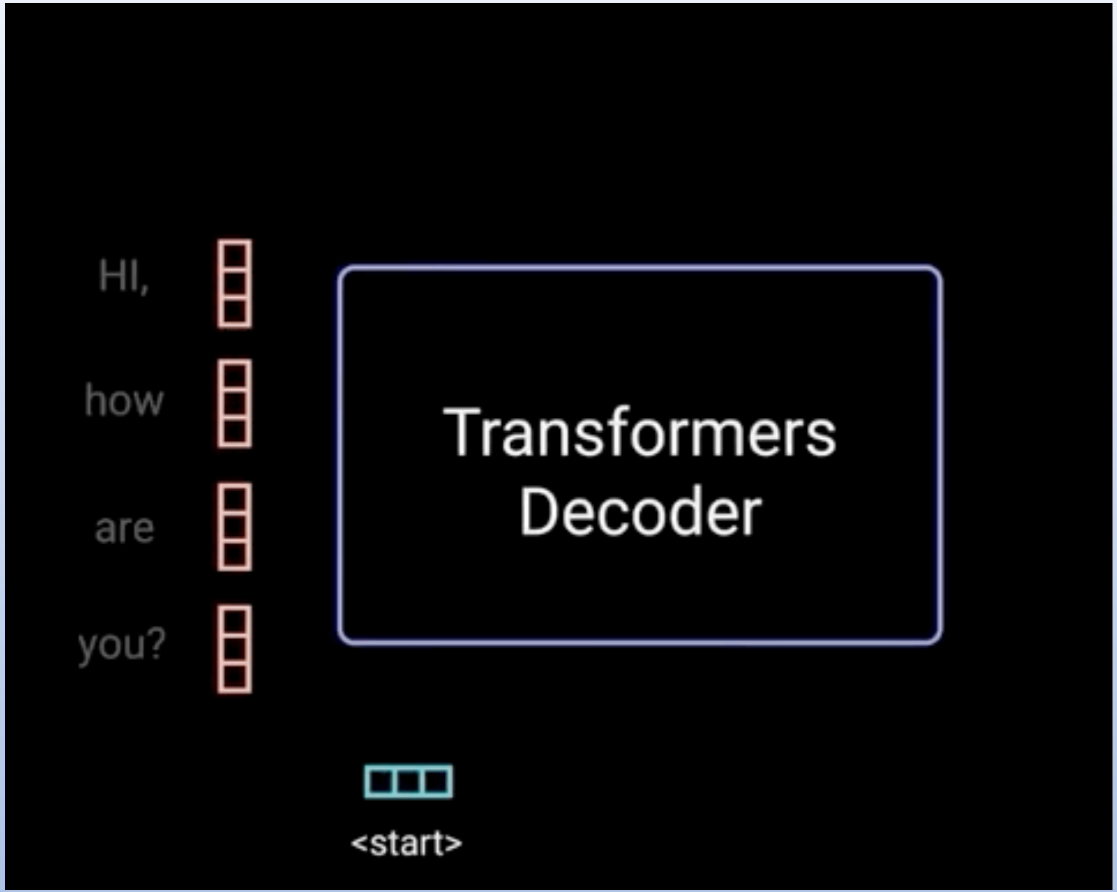
Mask is used in decoder to exclude words/tokens that haven’t appeared in the target.

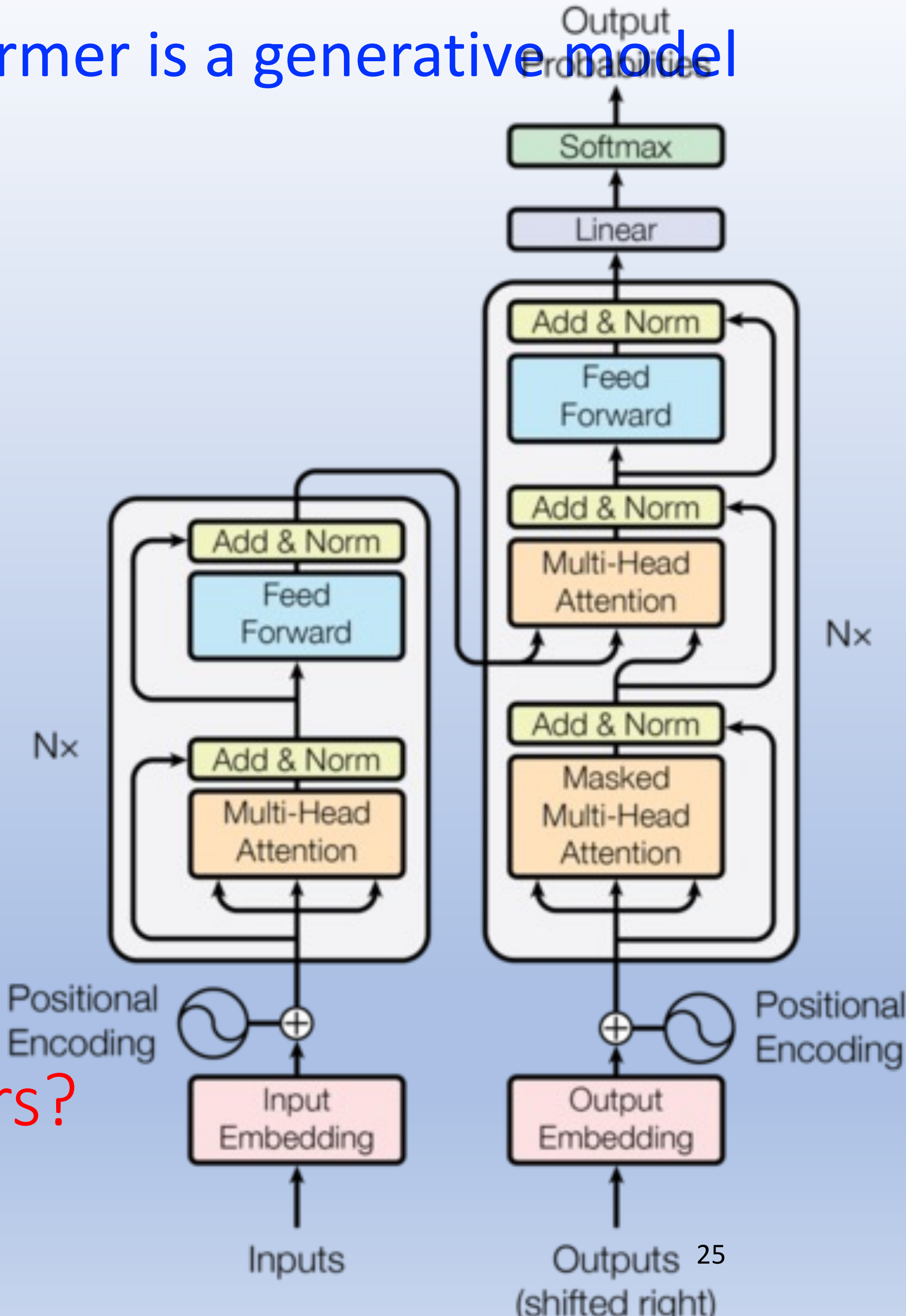


Encoder-Decoder Attention

Target sentence paying attention to source sentence

Inputs: Hi, How are you? Output: I am fine. Transformer is a generative model





Attention is the core component.

Where are the machine learnable parameters?

How are they learned and applied? CNN?

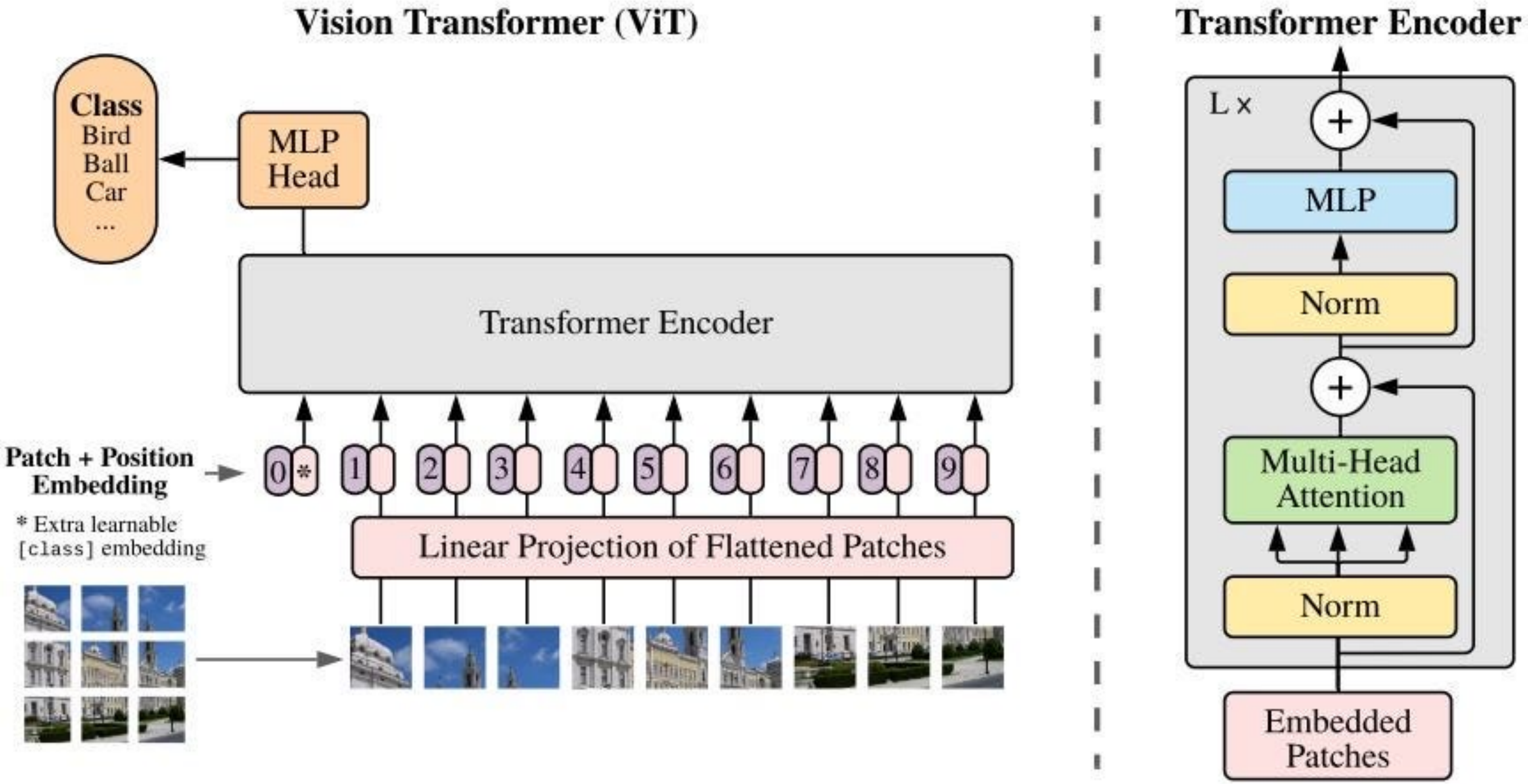


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification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

Vision Transformer vs CNN in Computer Vision

- The Vision Transformer (ViT) outperforms state-of-the-art convolutional networks in multiple benchmarks of computer vision while requiring fewer computational resources to train, after being pre-trained on **large amounts of data**.

- While CNNs have a proven track record in various computer vision tasks and handle large-scale datasets efficiently, Vision Transformers offer advantages in scenarios where global dependencies and contextual understanding are crucial.

Conclusion of Deep Learning

- Artificial Intelligence (AI) is booming because of the machine learning.
- Machine learning is booming because of the deep learning.
- Deep learning is booming thanks to the Convolutional Neural Networks (CNN).
- CNN is a very strongly regularized NN. Local attention/correlation/relation of CNN has its merits and limitations.
- Transformer is again a very strongly regularized NN. It performs all possible (global) specific attention/correlation/relation. It also applies convolution concept!
- The most critical of machine learning is not simply to let machine to learn from the big data, but to use human knowledge to guide (regularize) the machine to learn the data.
- This is the central task of all computer vision and machine learning problems!