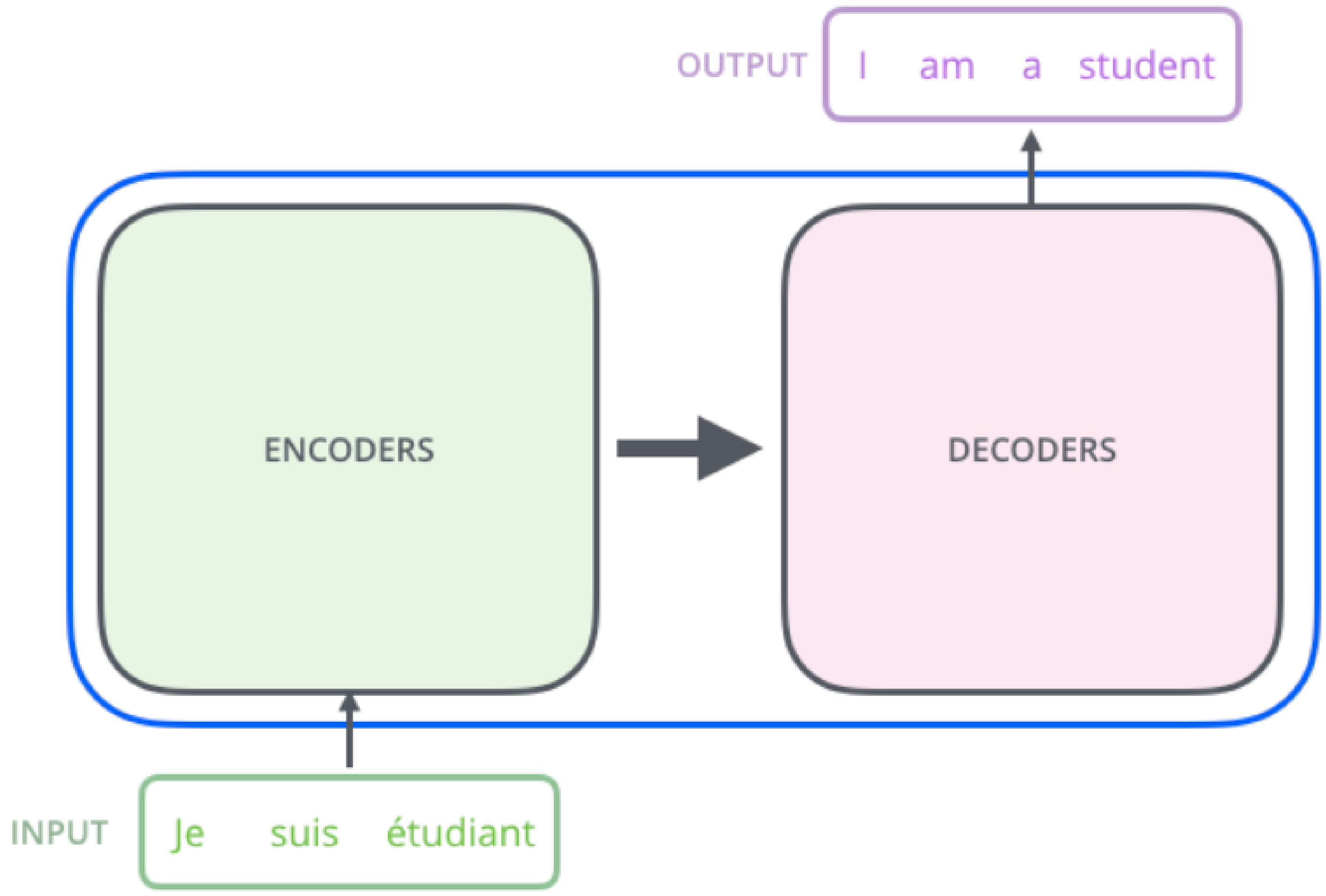
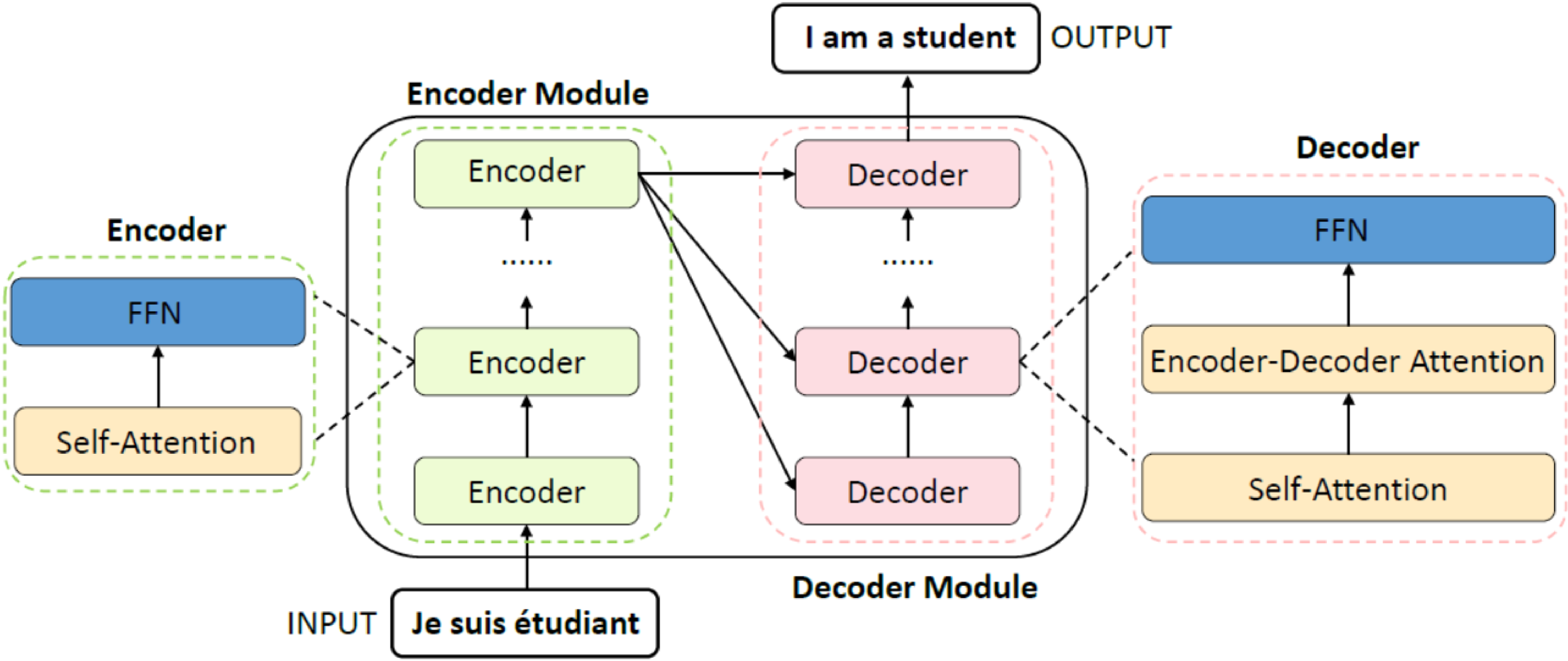
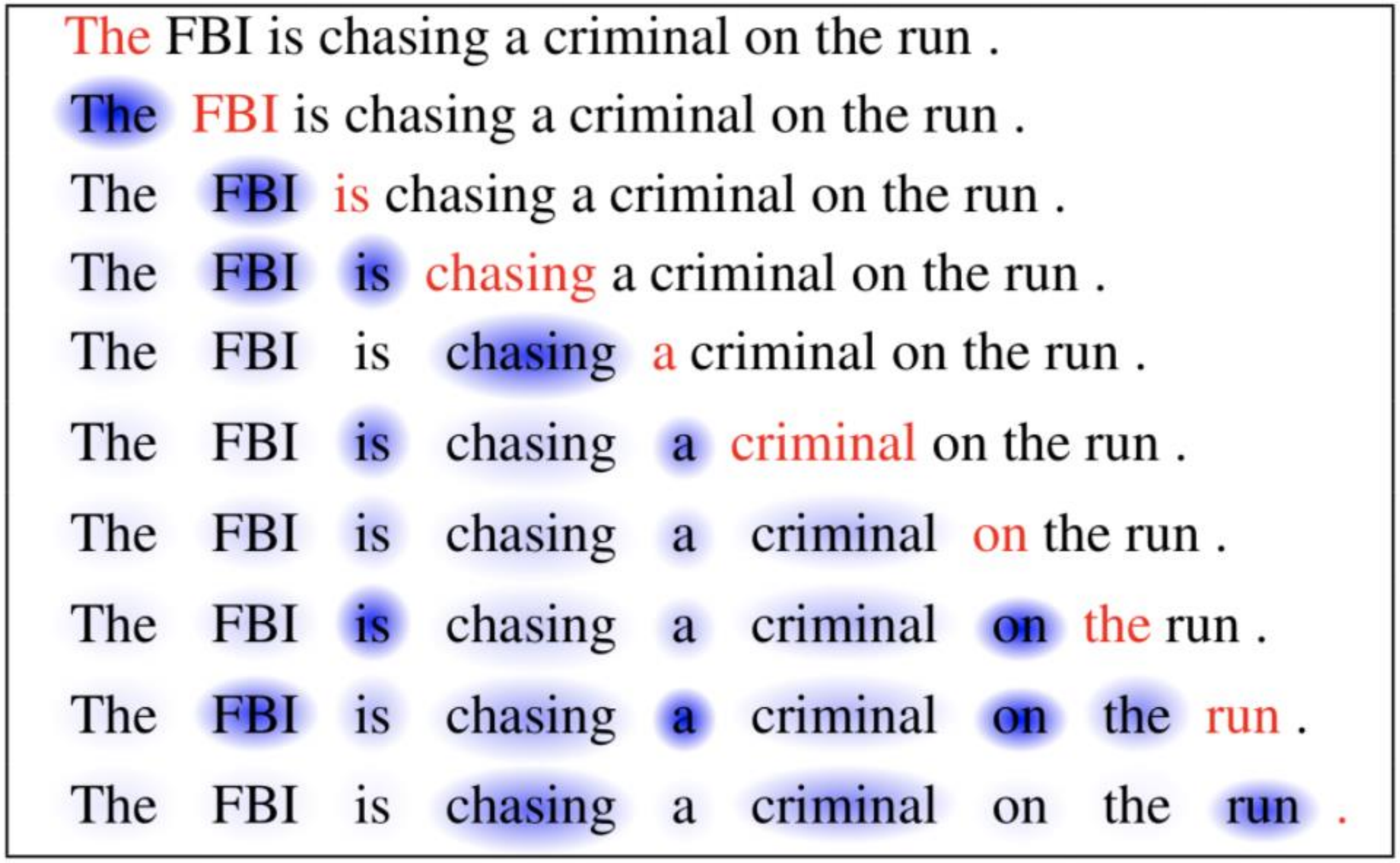
Transformer及在CV领域的应用


OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant



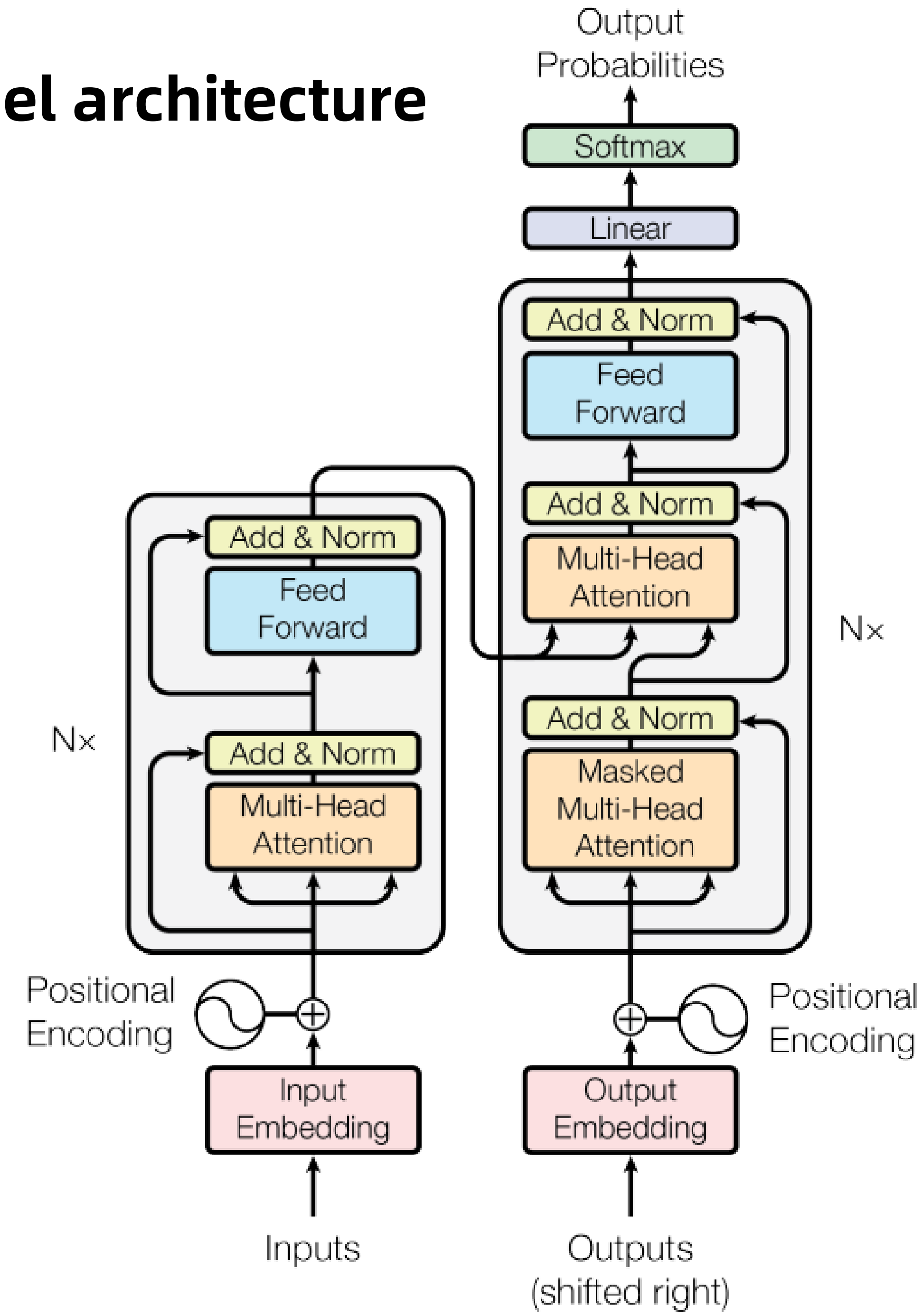
Pipeline of vanilla transformer.

Self-Attention

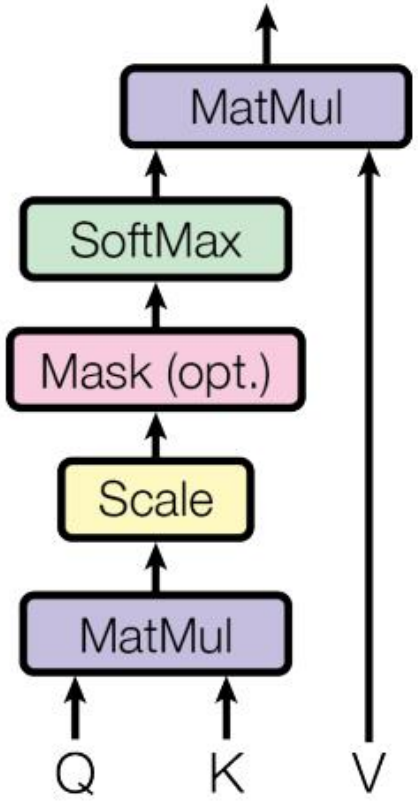


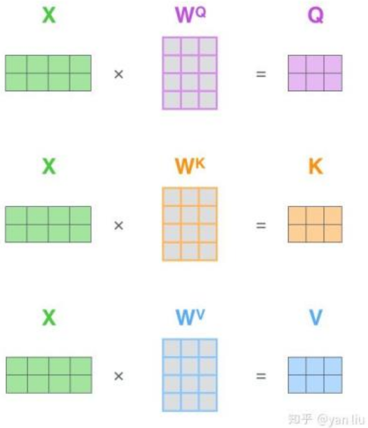
The current word is in red and the size of the blue shade indicates the activation level.

Transformer - model architecture



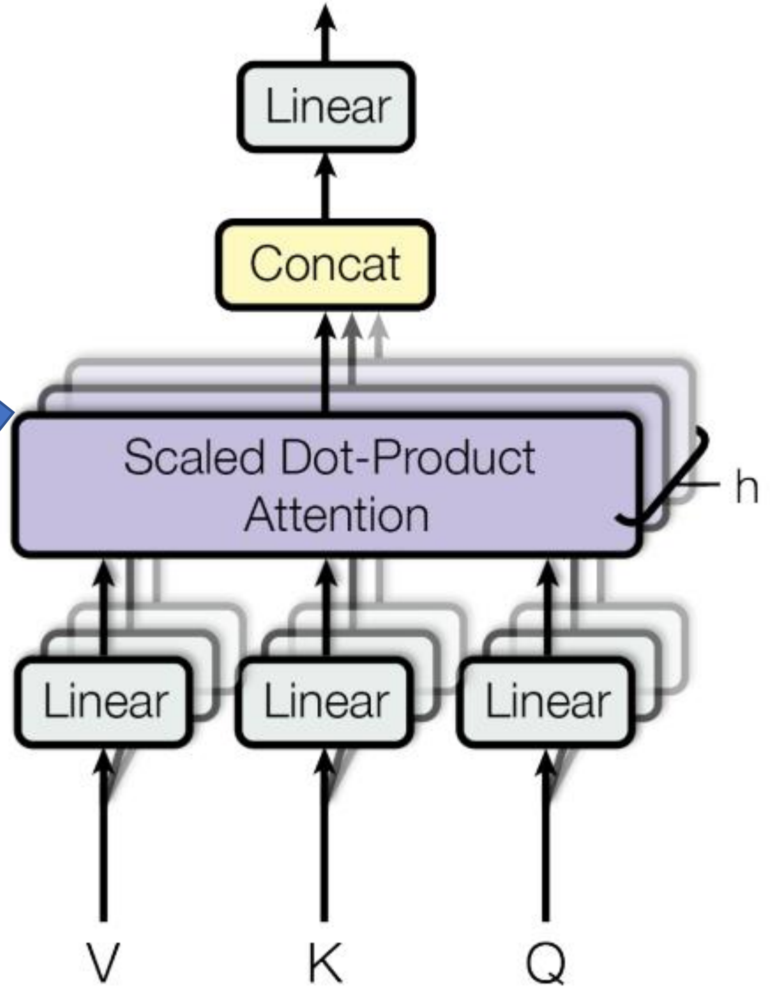
Scaled Dot-Product Attention





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

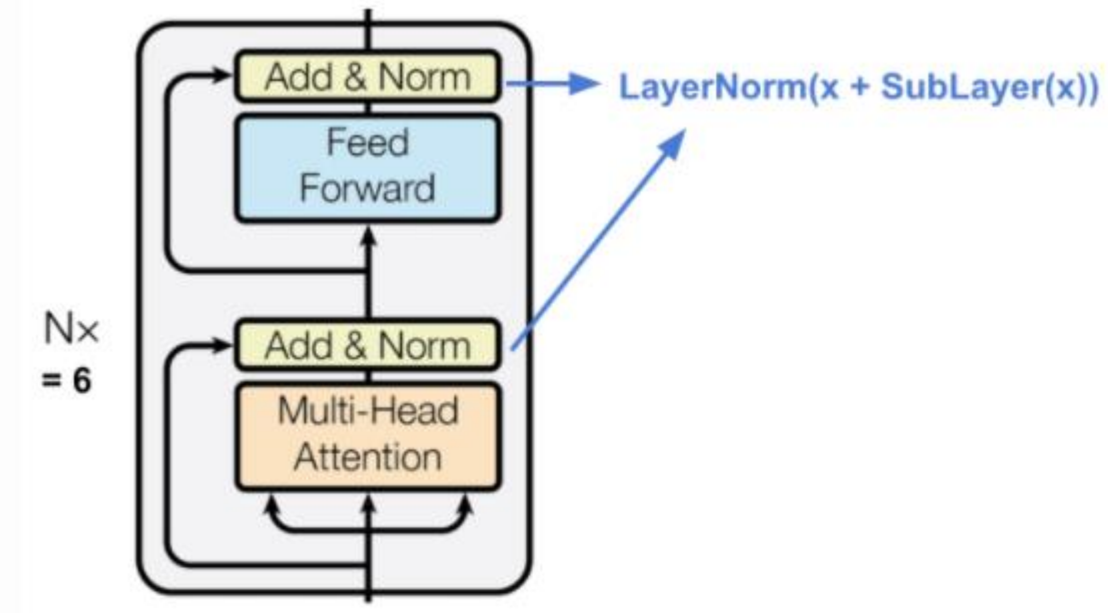
Multi-Head Attention



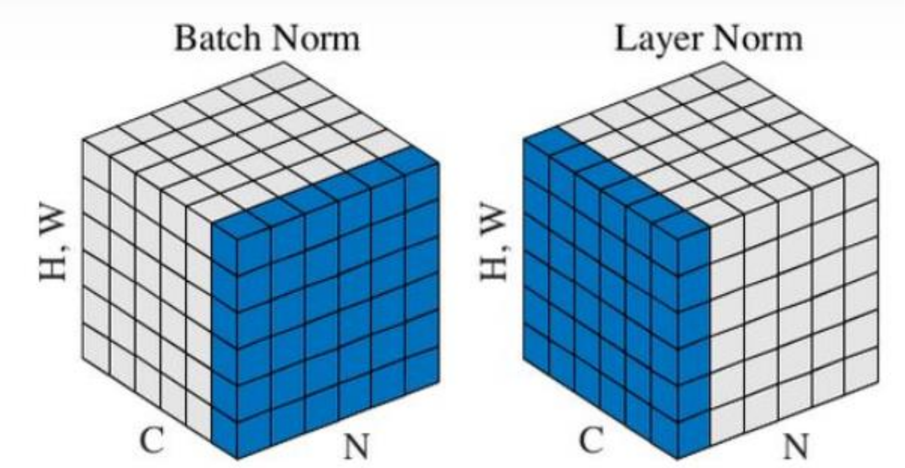
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\ldots,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Other Key Concepts in Transformer

Residual Connection in the Encoder and Decoder



$$\mathrm{LayerNorm}(\mathbf{X} + \mathrm{Attention}(\mathbf{X})).$$



Here, X is used as the input of self-attention layer, and the query, key and value matrices Q, K and V are all derived from the same input matrix X.

Feed-Forward Network

This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW1 + b1)W2 + b2$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer.

Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_f = 2048$.

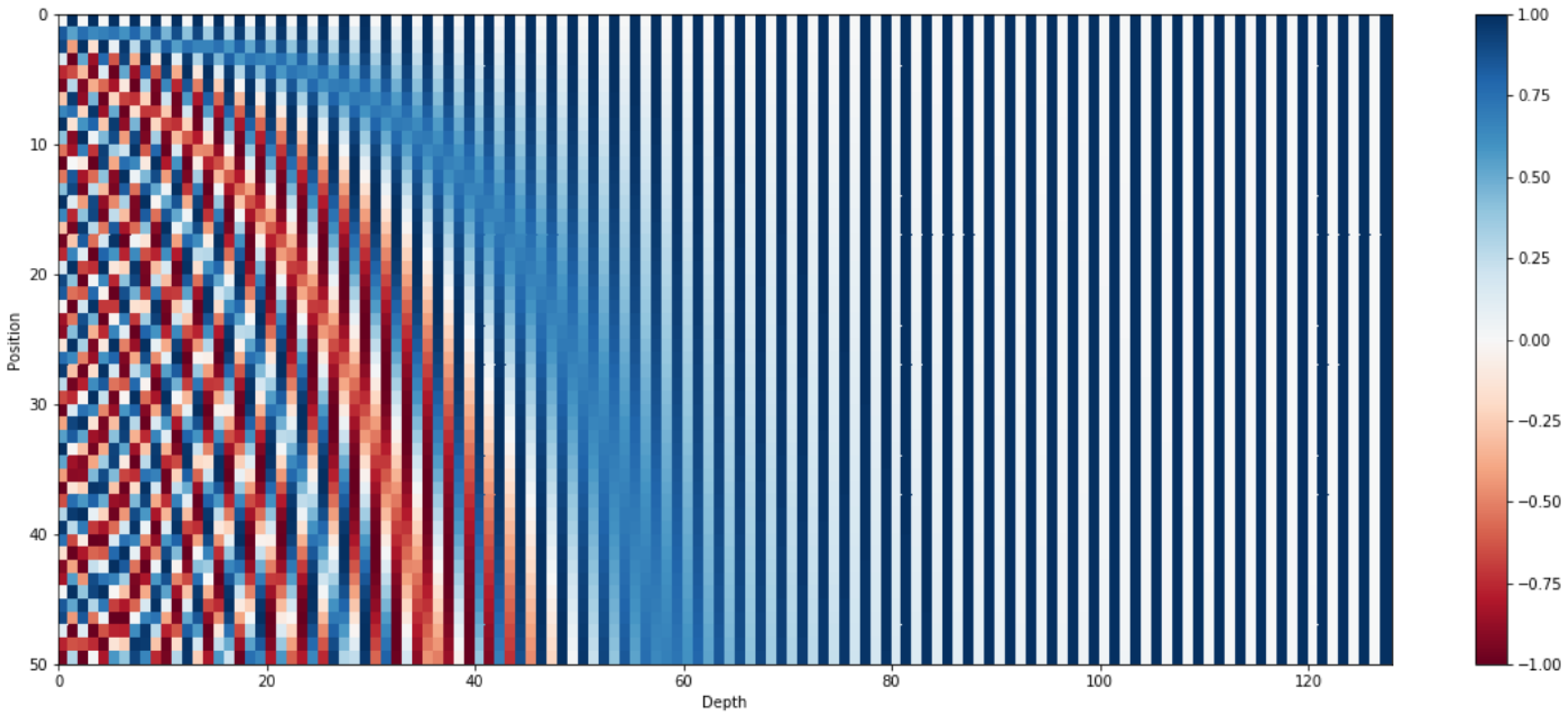
Positional Encoding

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

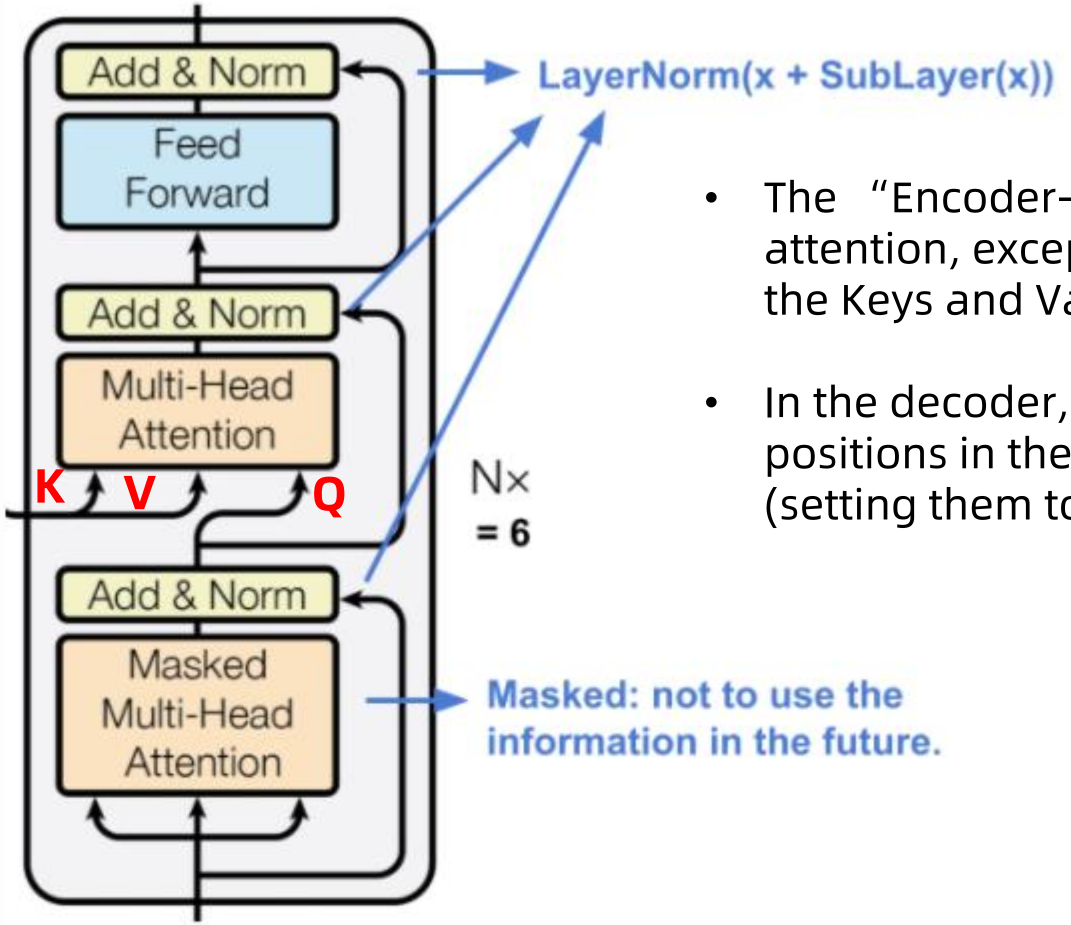
where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

Positional Encoding

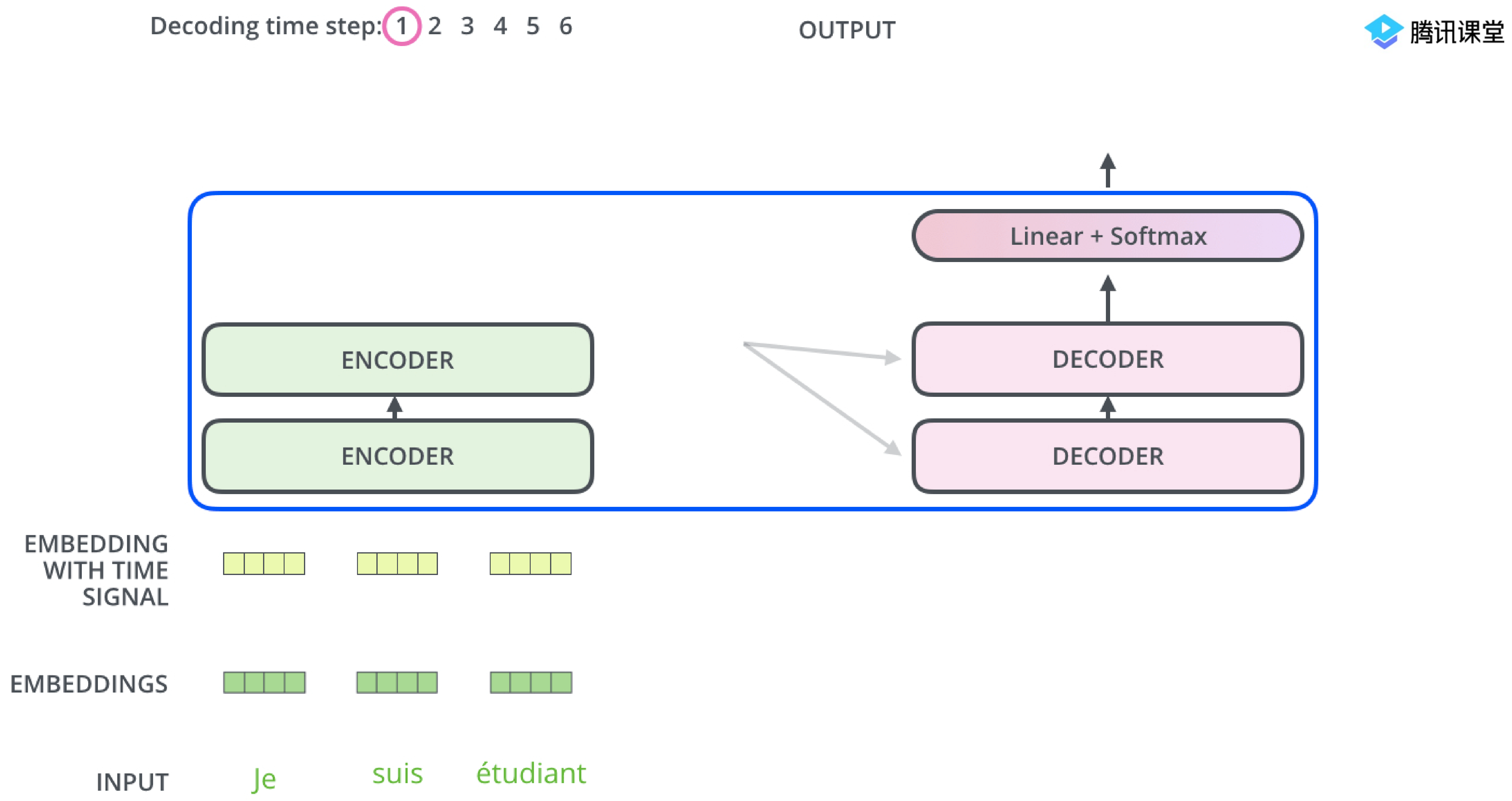


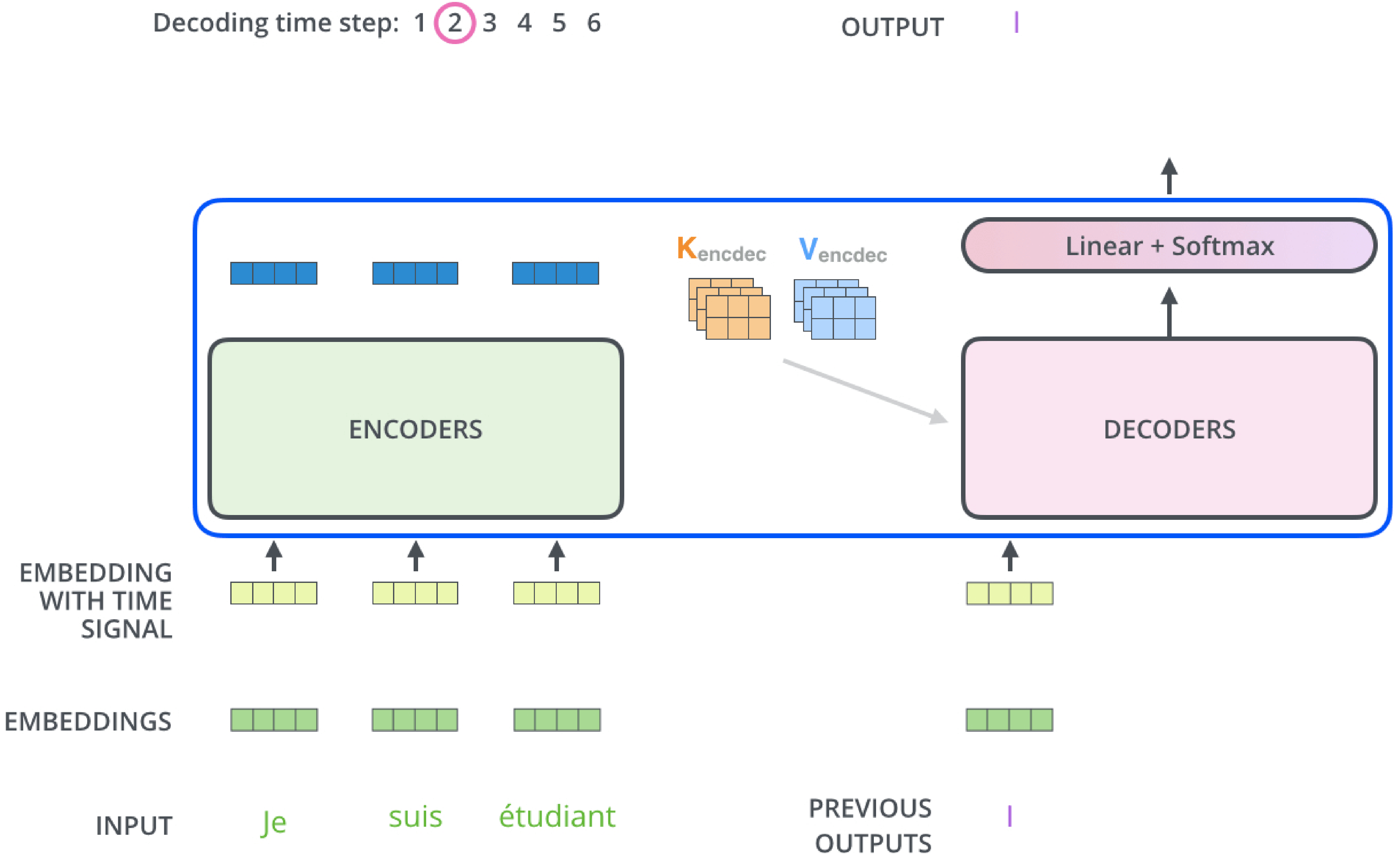
The 128-dimensional positonal encoding for a sentence with the maximum lenght of 50. Each row represents the embedding vector $\overrightarrow{p_t}$

Decoder



- The "Encoder-Decoder Attention" layer works just like multiheaded self-attention, except it creates its Queries matrix from the layer below it, and takes the Keys and Values matrix from the output of the encoder stack.
- In the decoder, the self-attention layer is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions (setting them to -inf) before the softmax step in the self-attention calculation.


Decoding time step: 1 2 3 4 5 6
OUTPUT
腾讯课堂
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Key milestones in the development of transformer

2017.6 | Transformer
Solely based on attention mechanism, the Transformer is proposed and shows great performance on NLP tasks.

2018.10 | BERT
Pre-training transformer models begin to be dominated in the field of NLP.

2020.5 | GPT-3
A huge transformer with 170B parameters, takes a big step towards general NLP model.

2020.5 | DETR
A simple yet effective framework for high-level vision by viewing object detection as a direct set prediction problem.

2020.7 | iGPT
The transformer model for NLP can also be used for image pre-training.

2020.10 | ViT
Pure transformer architectures work well for visual recognition.

2020.12 | IPT
The first transformer model for low-level vision by combining multi-tasks.

2021 | ViT Variants
Variants of ViT models, e.g., DeiT, PVT, TNT, and Swin.

TABLE 1: Representative works of vision transformers.

Category	Sub-category	Method	Highlights	Publication
Backbone	Supervised pretraining	ViT [55]	Image patches, standard transformer	ICLR 2021
		DeiT [219]	Data-efficient training, ViT model	ICML 2021
		Swin [17]	Shifted window, window-based self-attention	ICCV 2021
	Self-supervised pretraining	iGPT [29]	Pixel prediction self-supervised learning, GPT model	ICML 2020
		MoCo v3 [32]	Contrastive self-supervised learning, ViT	ICCV 2021
High/Mid-level vision	Object detection	DETR [19]	Set-based prediction, bipartite matching, transformer	ECCV 2020
		Deformable DETR [291]	DETR, deformable attention module	ICLR 2021
		ACT [284]	Adaptive clustering transformer	arXiv 2020
		UP-DETR [49]	Unsupervised pre-training, random query patch detection	CVPR 2021
		TSP [210]	New bipartite matching, encoder-only transformer	arXiv 2020
	Segmentation	Max-DeepLab [228]	PQ-style bipartite matching, dual-path transformer	CVPR 2021
		VisTR [235]	Instance sequence matching and segmentation	CVPR 2021
		SETR [285]	sequence-to-sequence prediction, standard transformer	CVPR 2021
	Pose Estimation	Hand-Transformer [102]	Non-autoregressive transformer, 3D point set	ECCV 2020
		HOT-Net [103]	Structured-reference extractor	MM 2020
		METRO [138]	Progressive dimensionality reduction	CVPR 2021

DETR

End to End Object Detection with Transformers

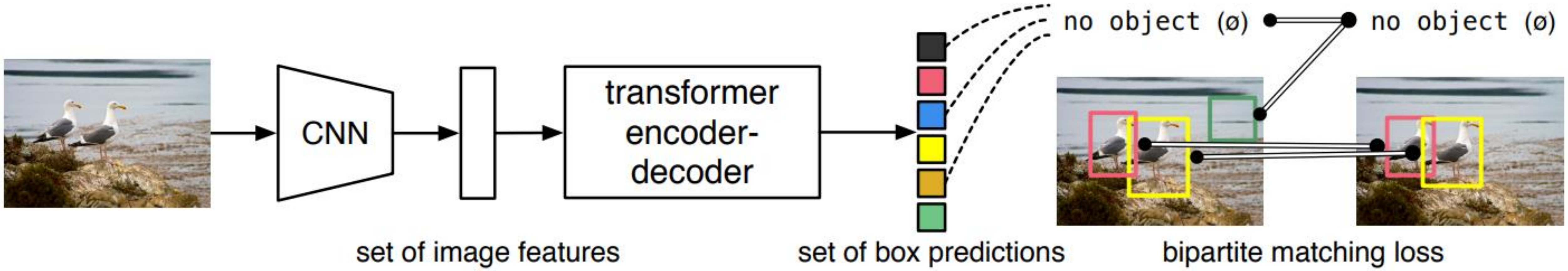
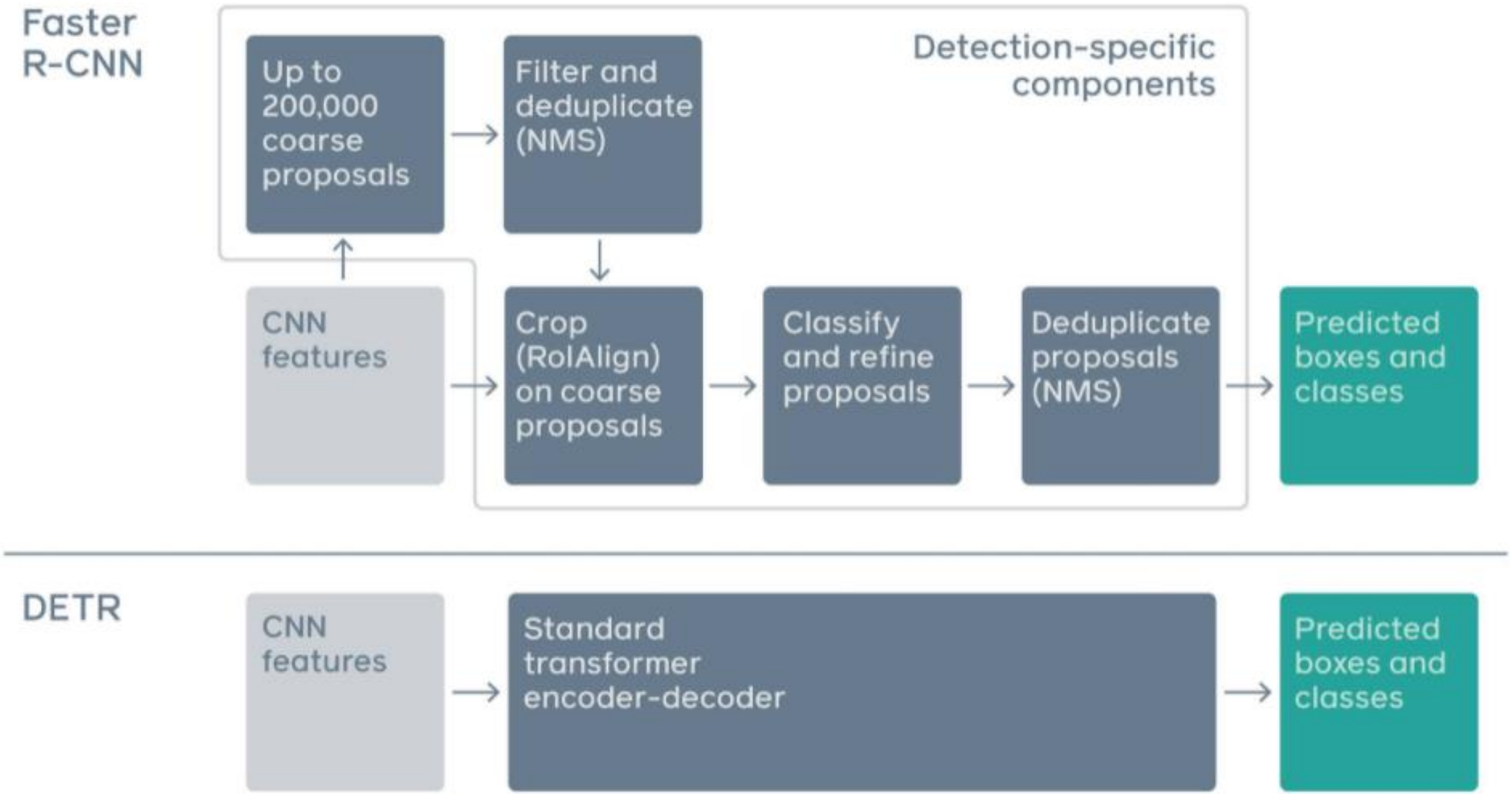


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.


Faster
R-CNN
Detection-specific
components
Up to
200,000
coarse
proposals
Filter and
deduplicate
(NMS)
CNN
features
Crop
(RoIAlign)
on coarse
proposals
Classify
and refine
proposals
Deduplicate
proposals
(NMS)
Predicted
boxes and
classes
DETR
CNN
features
Standard
transformer
encoder-decoder
Predicted
boxes and
classes

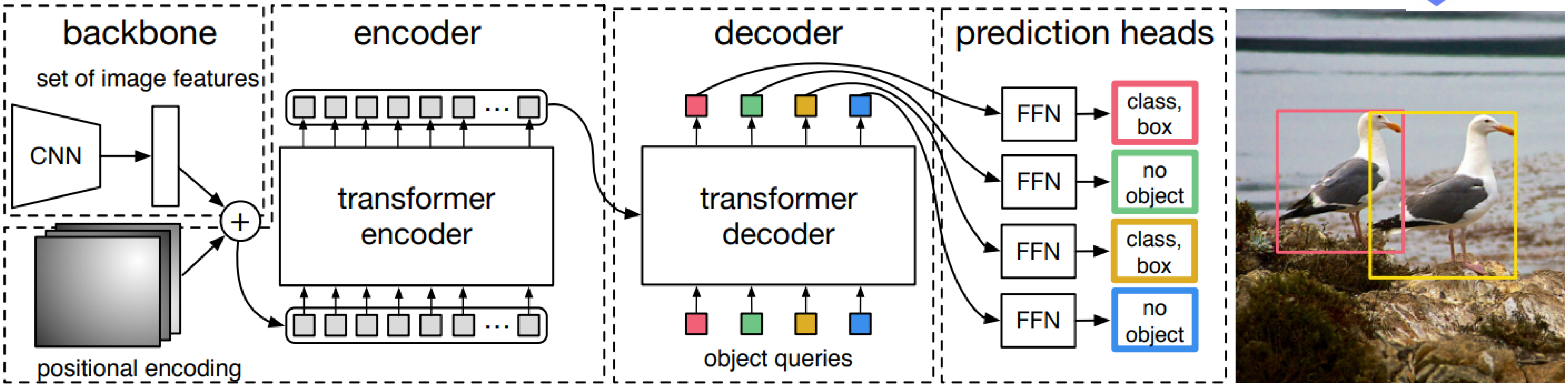
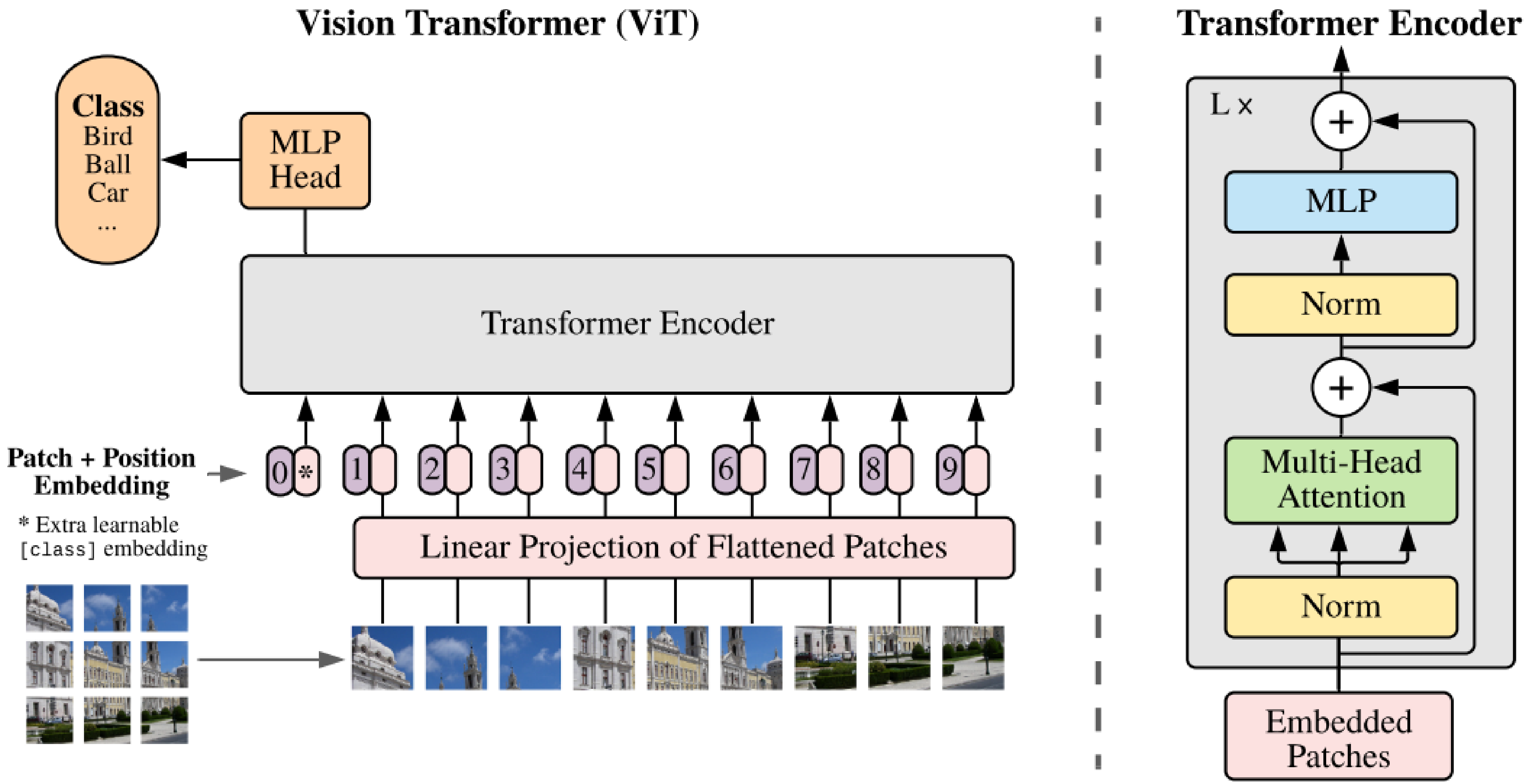


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.



Alexey Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021

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).

The MLP contains two layers with a GELU non-linearity.

$$
\begin{aligned}
\mathbf{z}_0 &= [\mathbf{x}_\text{class};\, \mathbf{x}^1_p\mathbf{E};\, \mathbf{x}^2_p\mathbf{E};\cdots;\, \mathbf{x}^N_p\mathbf{E}] + \mathbf{E}_{pos}, && \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} && (1)\\
\mathbf{z'}_\ell &= \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, && \ell = 1\ldots L && (2)\\
\mathbf{z}_\ell &= \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, && \ell = 1\ldots L && (3)\\
\mathbf{y} &= \mathrm{LN}(\mathbf{z}^0_L) && && (4)
\end{aligned}
$$