Image Processing and Analysis

Attention models
Transformers

olivier.bernard@insa-lyon.fr

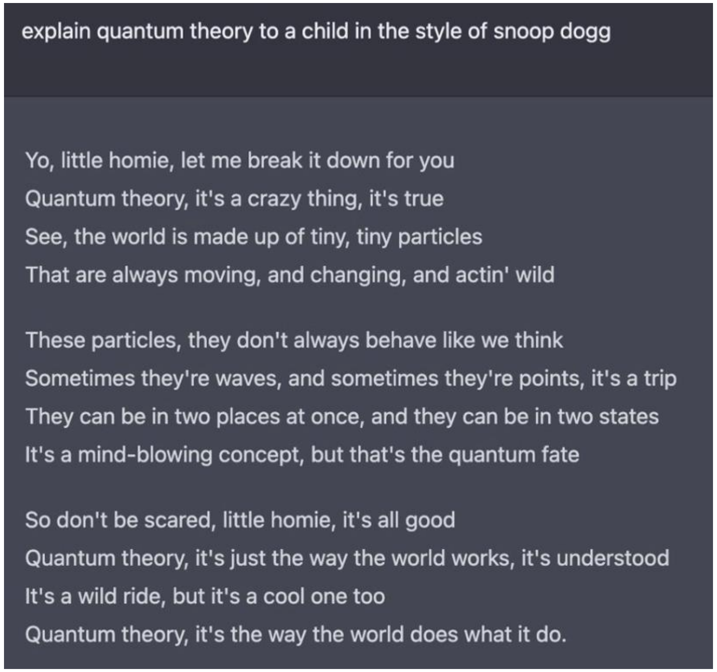
nicolas.thome@sorbonne-universite.fr

What are the applications of attention models?

- NLP: Natural Language Processing

Enable machines to understand, interpret, and generate natural language effectively

- Developed for tasks such as
  - Automatic translation
  - Speech comprehension
  - Text generation
  - Computer code generation
- Known algorithms
  - BERT
  - GPT 3, 4
  - Chat-GPT
  - [...]



explain quantum theory to a child in the style of snoop dogg

Yo, little homie, let me break it down for you
Quantum theory, it's a crazy thing, it's true
See, the world is made up of tiny, tiny particles
That are always moving, and changing, and actin' wild

These particles, they don't always behave like we think
Sometimes they're waves, and sometimes they're points, it's a trip
They can be in two places at once, and they can be in two states
It's a mind-blowing concept, but that's the quantum fate

So don't be scared, little homie, it's all good
Quantum theory, it's just the way the world works, it's understood
It's a wild ride, but it's a cool one too
Quantum theory, it's the way the world does what it do.

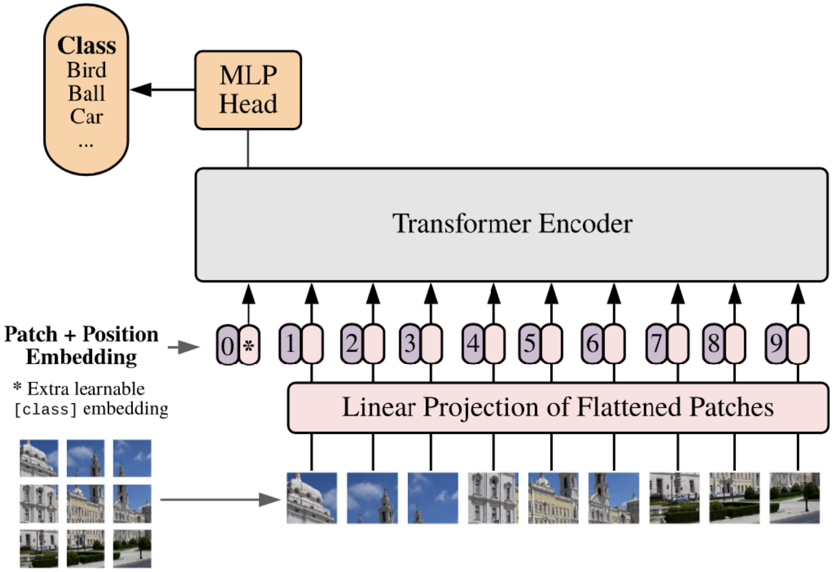
What are the applications of attention models?

- Also developed in image analysis

Enable machines to understand, analyze, and generate images efficiently

- State-of-the-art algorithms
  - Classification
  - Segmentation
  - Segmentation + time
  - Tracking
  - Image generation

Vision Transformer - 2020


Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

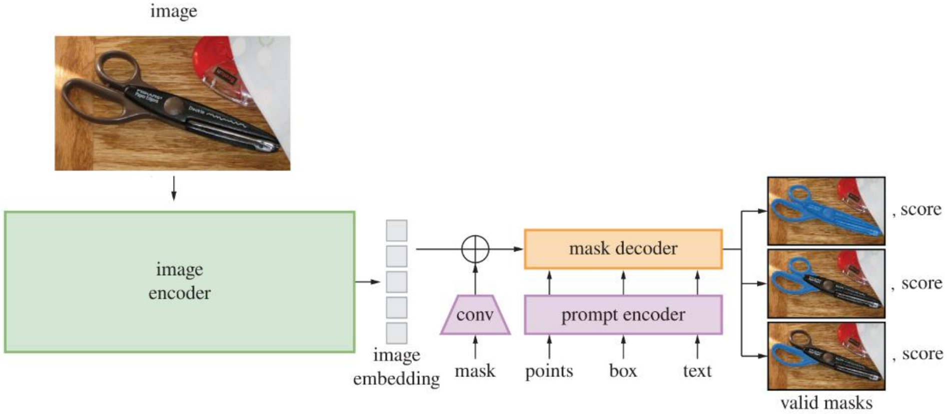
What are the applications of attention models?

- Also developed in image analysis

Enable machines to understand, analyze, and generate images efficiently

- State-of-the-art algorithms
  - Classification
  - Segmentation
  - Segmentation + time
  - Tracking
  - Image generation

SAM (Segment Anything) - 2024



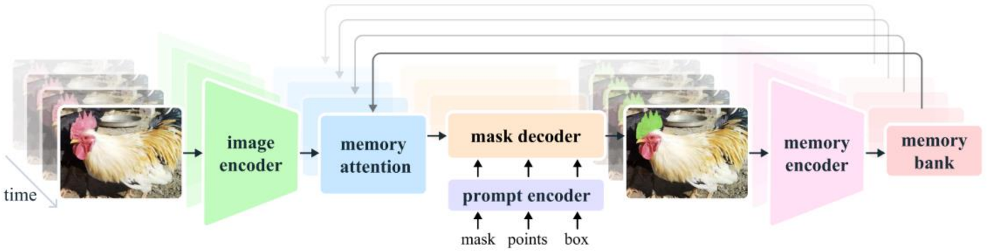
What are the applications of attention models?

- Also developed in image analysis

Enable machines to understand, analyze, and generate images efficiently

- State-of-the-art algorithms
  - Classification
  - Segmentation
  - Segmentation + time
  - Tracking
  - Image generation

SAM2 - 2024



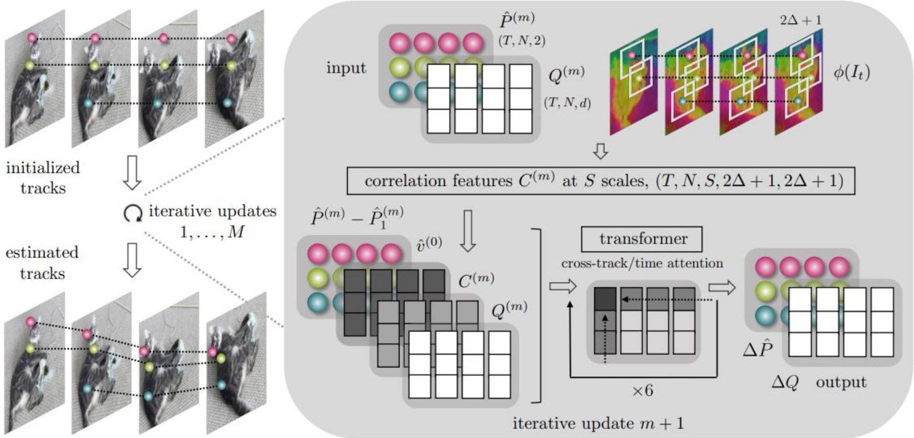
What are the applications of attention models?

- Also developed in image analysis

Enable machines to understand, analyze, and generate images efficiently

- State-of-the-art algorithms
  - Classification
  - Segmentation
  - Segmentation + time
  - Tracking
  - Image generation

CoTracker - 2024



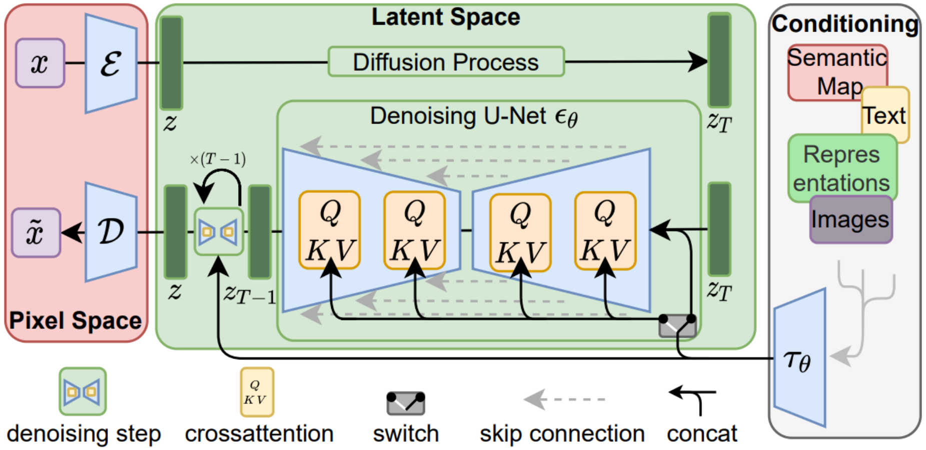
What are the applications of attention models?

- Also developed in image analysis

Enable machines to understand, analyze, and generate images efficiently

- State-of-the-art algorithms
  - Classification
  - Segmentation
  - Segmentation + time
  - Tracking
  - Image generation

Latent diffusion model - 2023



Transformers

Tokens

Tokens generation

- The input data is structured as *tokens*

→ Text: token = word of a sentence
→ Image: token = patch of an image

Text

« Hello, I am olivier »

Tokenization procedure

« Hello », « I », « am », « olivier »

« Hello » $\Rightarrow x_i \in \mathbb{R}^t$

0	**1**	**0**	**0**	**0**

Image

Tokenization procedure

$\Rightarrow \quad x_i \in \mathbb{R}^{P^2 C}$

P patch width, $C = 3$ if color image

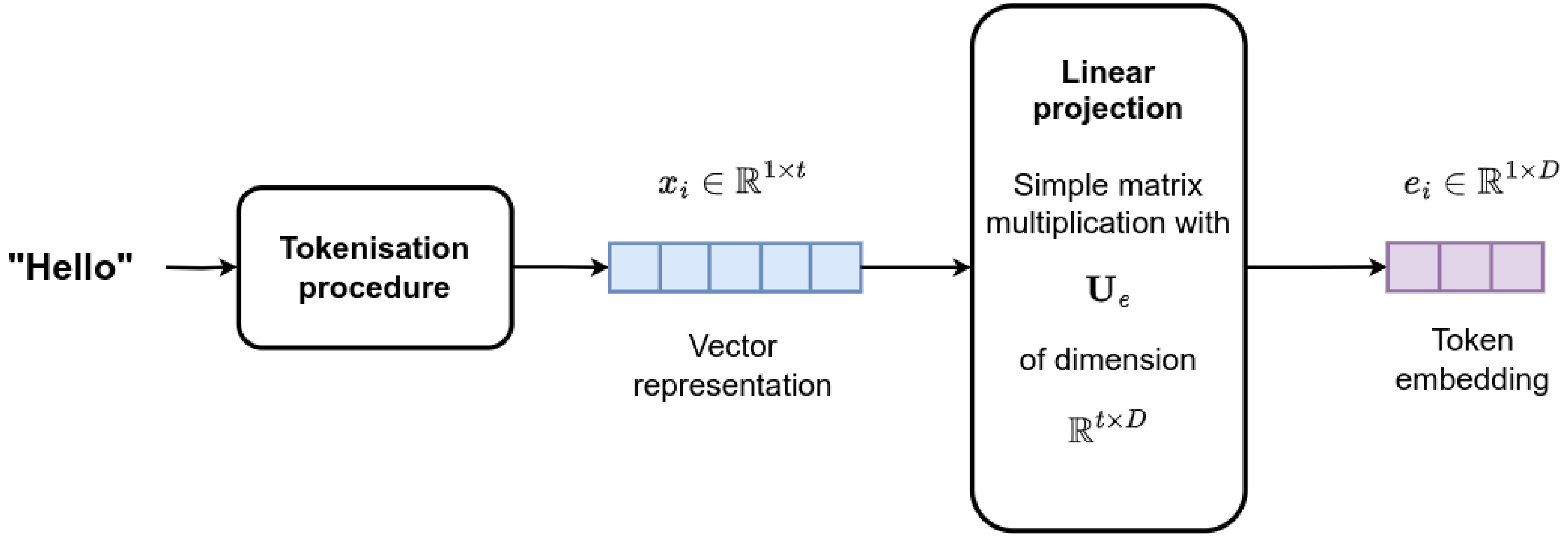
Tokens generation

- Creation of a representation (or *embedding*) of the tokens

  → Simple linear projection

  → Multiplication by a learnable representation matrix $\boldsymbol{U}_e$

$$\boxed{e_i = x_i\, \boldsymbol{U}_e}$$



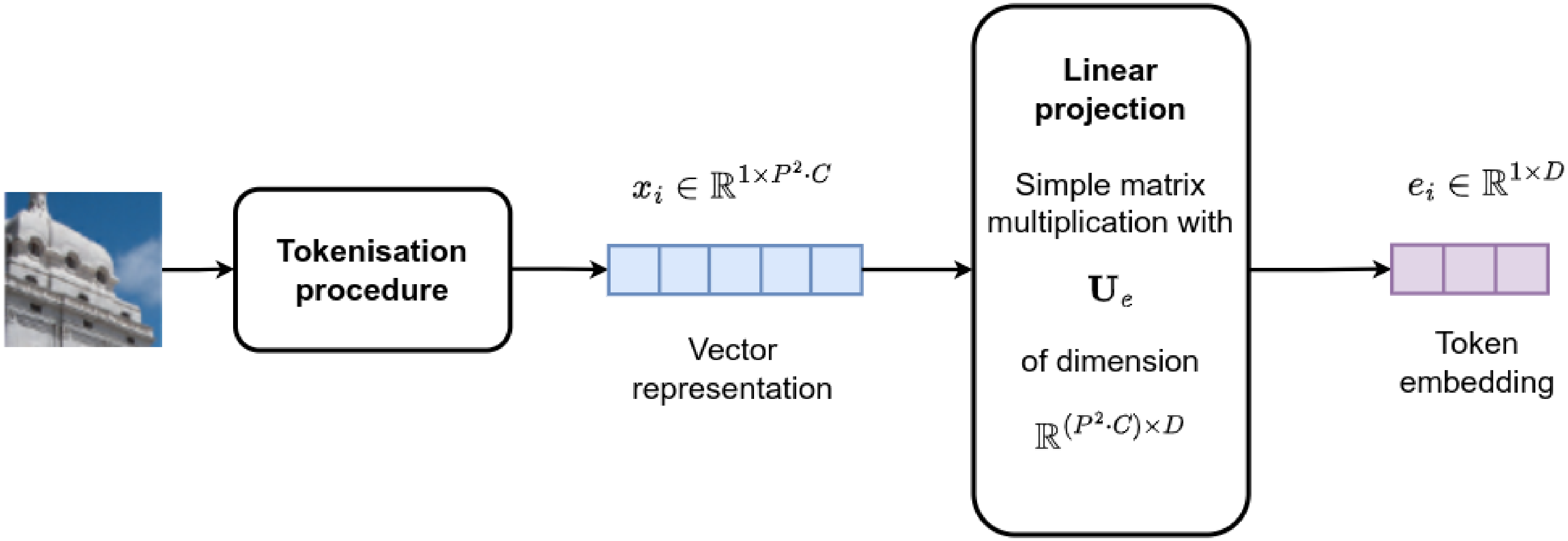
Tokens generation

- Creation of a representation (or *embedding*) of the tokens

  ➔ Simple linear projection

  ➔ Multiplication by a learnable representation matrix $\boldsymbol{U}_e$

$$\boxed{e_i = x_i\, \boldsymbol{U}_e}$$



Tokenisation procedure

$x_i \in \mathbb{R}^{1 \times P^2 \cdot C}$

Vector representation

Linear projection

Simple matrix multiplication with

$\mathbf{U}_e$

of dimension

$\mathbb{R}^{(P^2 \cdot C) \times D}$

$e_i \in \mathbb{R}^{1 \times D}$

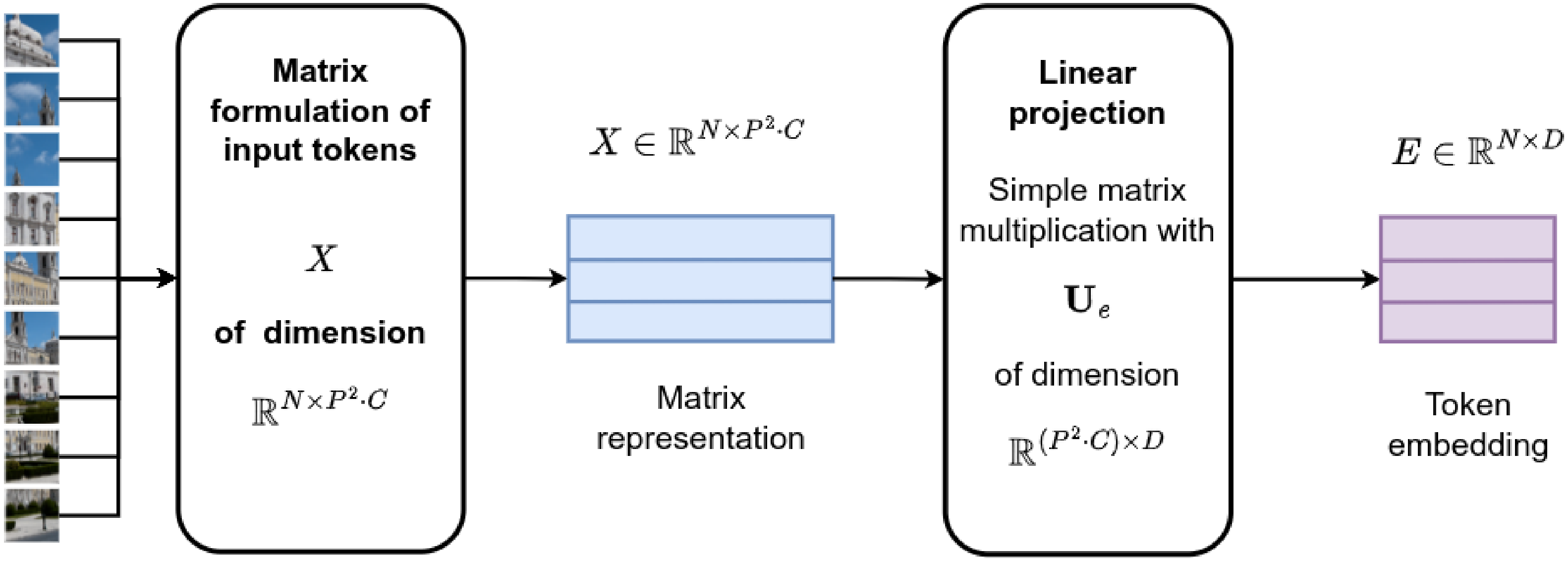
Token embedding

Tokens generation

- Learning a representation matrix $\boldsymbol{U}_e$ shared by each token

  ➔ Matrix formulation

$$\boxed{E = X\,\boldsymbol{U}_e}$$



Matrix formulation of input tokens

X

of dimension

$\mathbb{R}^{N\times P^2\cdot C}$

$X \in \mathbb{R}^{N\times P^2\cdot C}$

Matrix representation

Linear projection

Simple matrix multiplication with

$\mathbf{U}_e$

of dimension

$\mathbb{R}^{(P^2\cdot C)\times D}$

$E \in \mathbb{R}^{N\times D}$

Token embedding

Tokens generation

- Positional encoding
  - ➔ Sentence / image: a set of independent tokens
  - ➔ Loss of structural information from the input data
- Recovery of structure: positional encoding (PE: positional embedding)
  - ➔ Correspondence between the position of token t and a vector $p_t \,\epsilon\, \mathbb{R}^{1\times D}$
  - ➔ Classical encoding: sinusoidal function

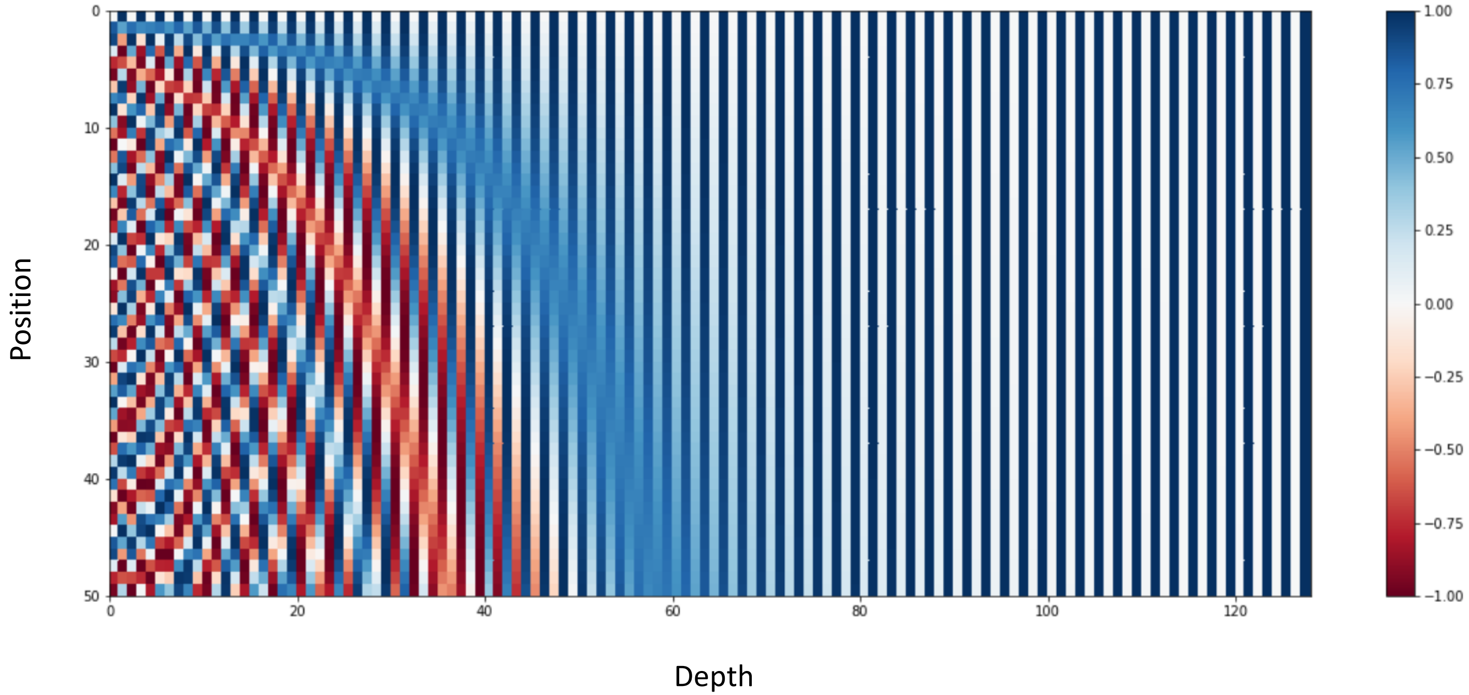
$$p_t \;\epsilon\; \mathbb{R}^{1\times D}$$

$$p_t = \left[\sin(\omega_1 t)\,,\cos(\omega_1 t)\,,\cdots,\sin\left(\omega_{D/2} t\right),\cos\left(\omega_{D/2} t\right)\right]$$

$$\omega_k = \frac{1}{10000^{2k/D}}$$

Tokens generation

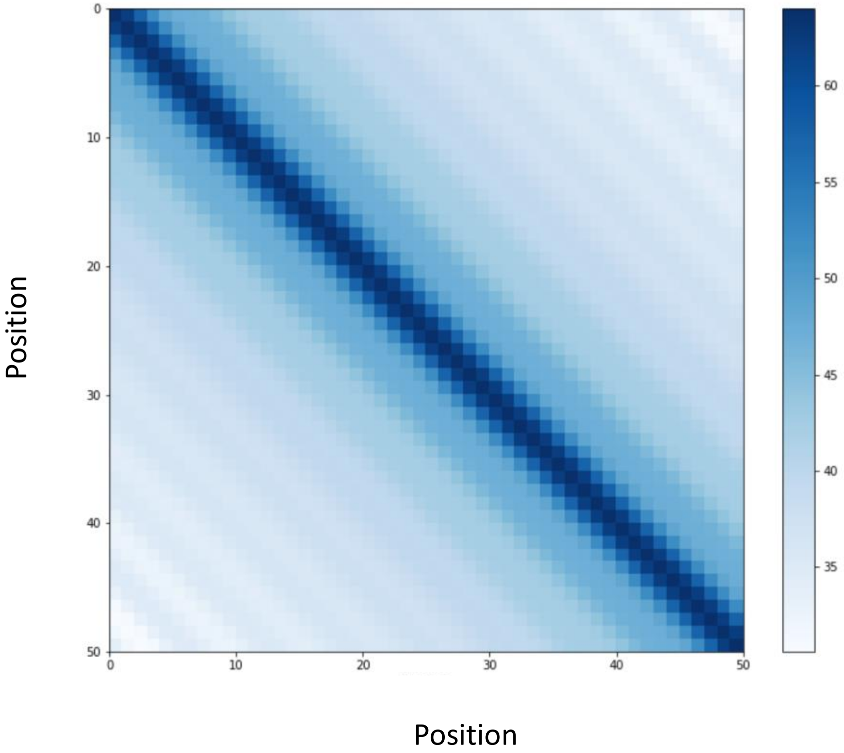
- Sinusoidal positional encoding
  - ➔ Unique vector p_t for each position t
  - ➔ $p_t\ (i)\ \in\ [-1{,}1]$: Intrinsic normalization of values



Number of tokens = 50, dimension D of each token = 128

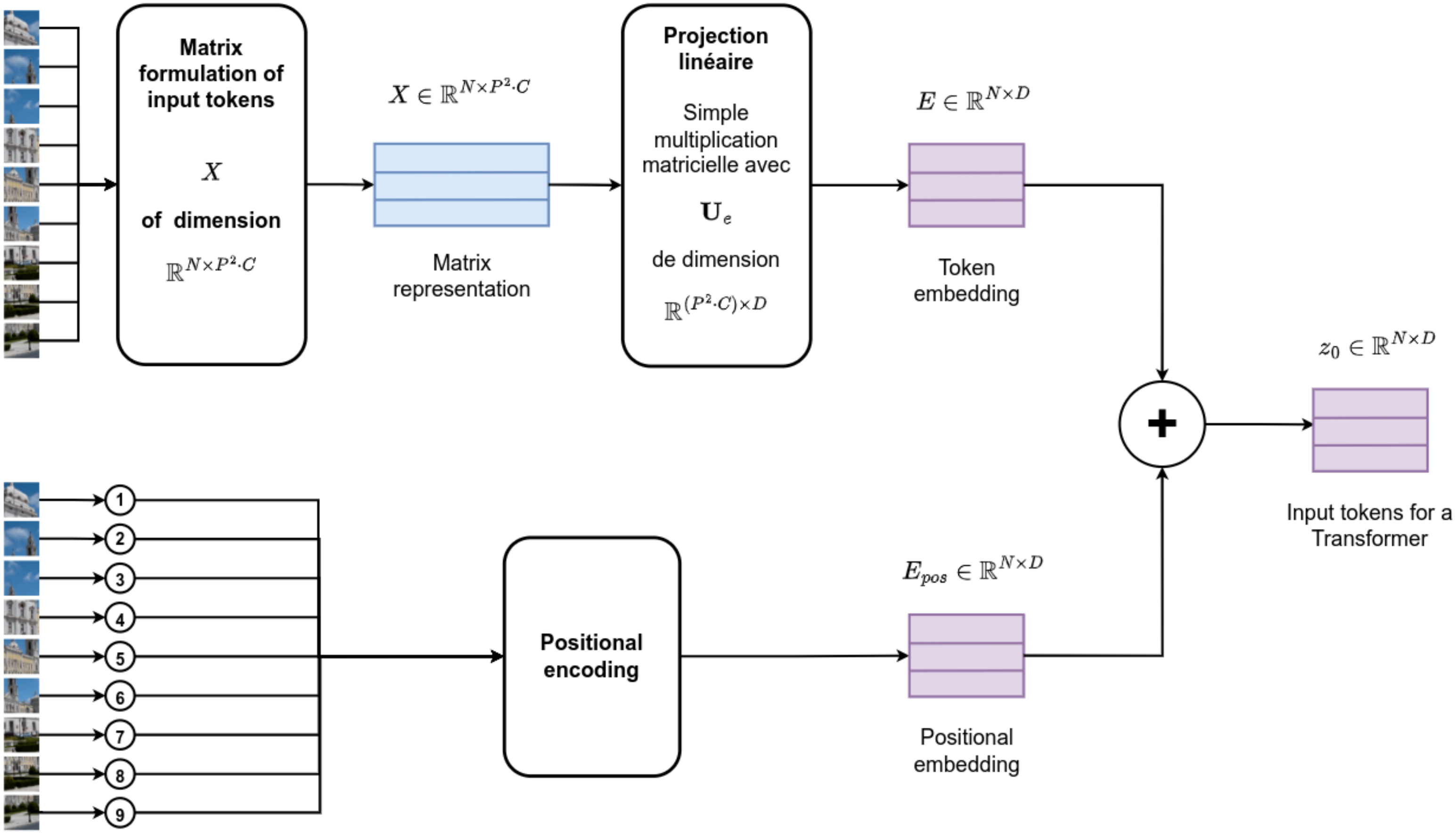
Tokens generation

- Sinusoidal positional encoding
  - Intrinsic modeling of the relative position of tokens
  - Position similarity matrix: $K = P \cdot P^t$



Tokens generation

- Final representation
  - → Final tokens = sum of token and position representations
  - → Only the matrix U_e is to be learned for this phase


Matrix formulation of input tokens
X
of dimension
$\mathbb{R}^{N \times P^2 \cdot C}$
$X \in \mathbb{R}^{N \times P^2 \cdot C}$
Matrix representation
Projection linéaire
Simple multiplication matricielle avec
$\mathbf{U}_e$
de dimension
$\mathbb{R}^{(P^2 \cdot C) \times D}$
$E \in \mathbb{R}^{N \times D}$
Token embedding
$z_0 \in \mathbb{R}^{N \times D}$
Input tokens for a Transformer
1
2
3
4
5
6
7
8
9
Positional encoding
$E_{pos} \in \mathbb{R}^{N \times D}$
Positional embedding

Transformers

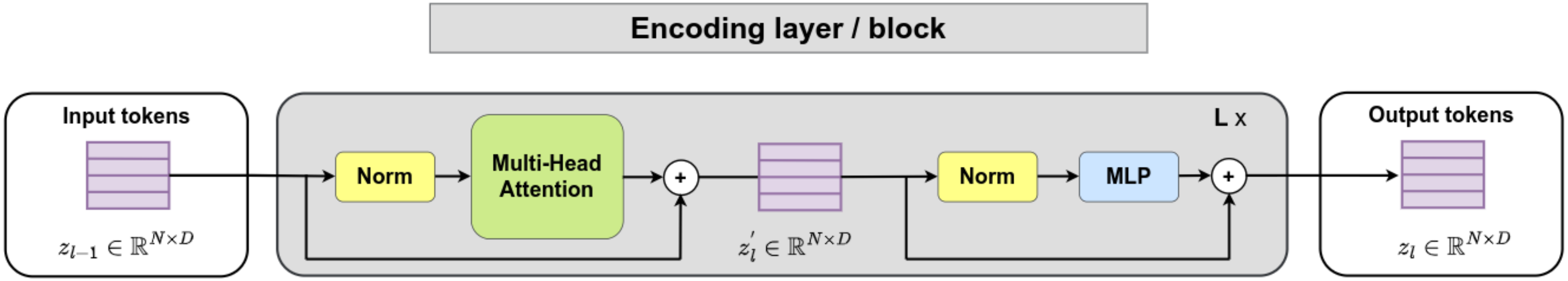
Encoding blocks

Transformer: information encoding

- Encoder

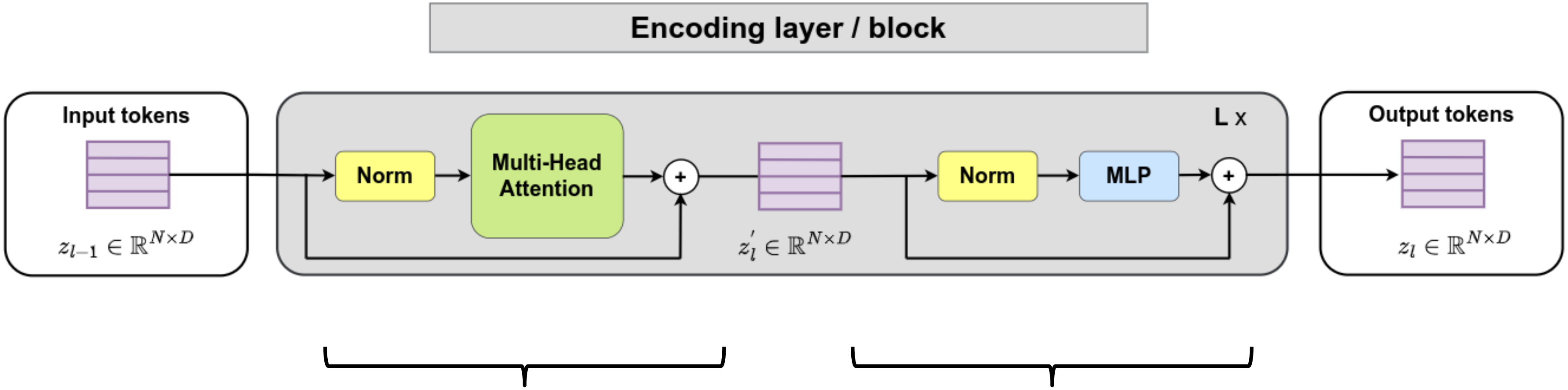
  ➔ Corresponds to N encoding blocks

  - Input: A token representation
  - Output: A new token representation tailored to the target is being optimized



Encoding layer / block
Input tokens
$z_{l-1} \in \mathbb{R}^{N \times D}$
L x
Norm
Multi-Head Attention
+
$z_l' \in \mathbb{R}^{N \times D}$
Norm
MLP
+
Output tokens
$z_l \in \mathbb{R}^{N \times D}$

Transformer: information encoding


Encoding layer / block
Input tokens
$z_{l-1} \in \mathbb{R}^{N \times D}$
Norm
Multi-Head Attention
+
$z_l' \in \mathbb{R}^{N \times D}$
L x
Norm
MLP
+
Output tokens
$z_l \in \mathbb{R}^{N \times D}$

1st step

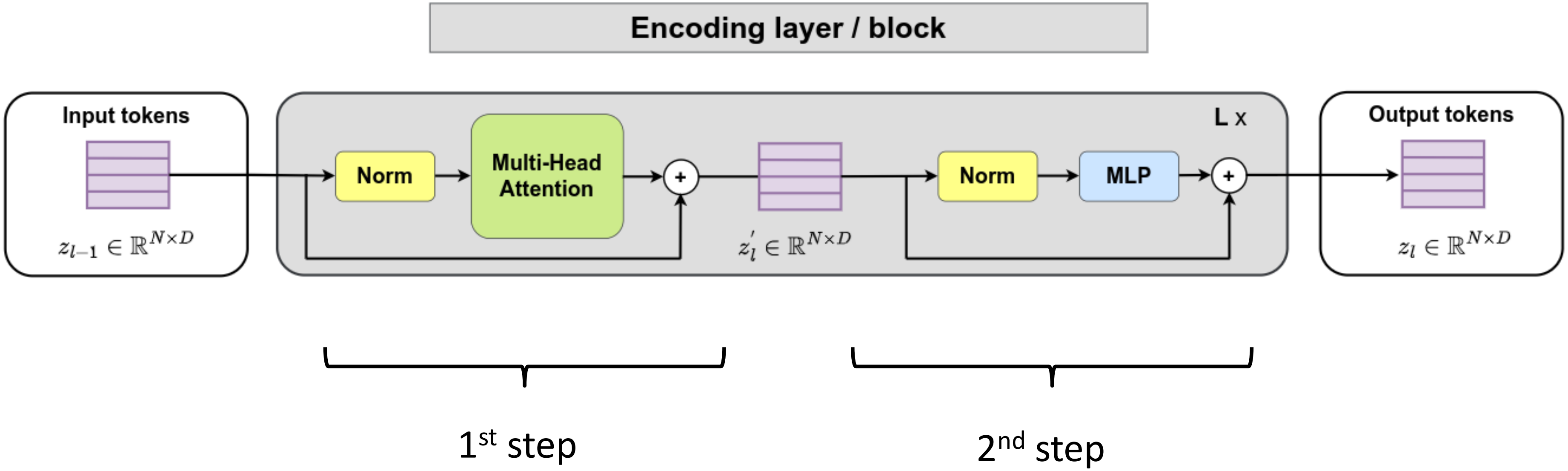
2nd step

- 1st step
  - ➔ Computation of attention maps between tokens
  - ➔ Residual connection 1) Against vanishing gradient

    2) Do not forget the positional representation

$$\boxed{z_l{}' = MHA\big(LN(z_{l-1})\big) + \ z_{l-1}}$$

Transformer: information encoding


Encoding layer / block
Input tokens
$z_{l-1} \in \mathbb{R}^{N \times D}$
Norm
Multi-Head Attention
+
$z_l' \in \mathbb{R}^{N \times D}$
Norm
MLP
+
L x
Output tokens
$z_l \in \mathbb{R}^{N \times D}$
1st step
2nd step

- 2nd step

→ Introduction of nonlinearities to generate relevant information

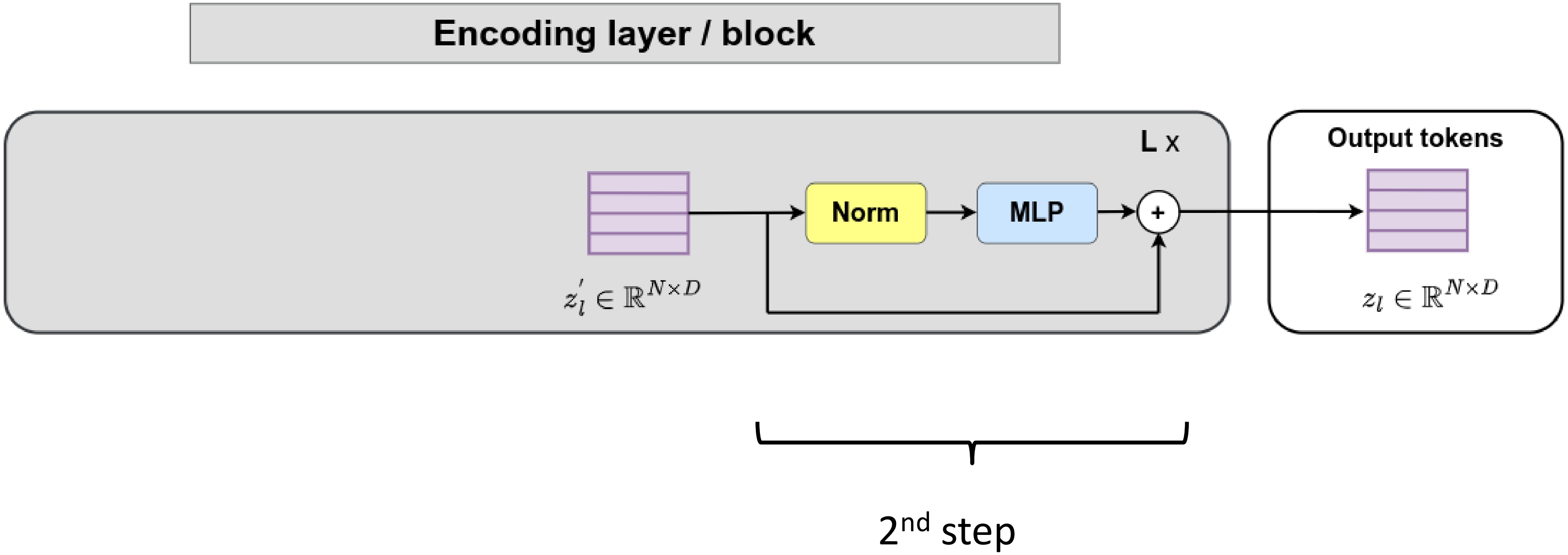
→ Residual connection 1) Against vanishing gradient

2) Do not forget the positional representation

$$z_l = MLP\big(LN(z_l')\big) + z_l'$$

Transformer: information encoding

- Focus on the 2nd step

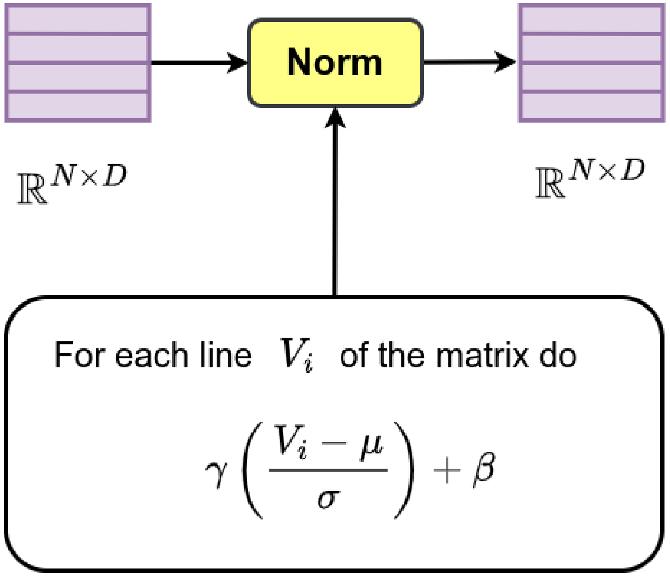

Encoding layer / block
L x
Norm
MLP
+
$z_l' \in \mathbb{R}^{N \times D}$
Output tokens
$z_l \in \mathbb{R}^{N \times D}$
2nd step

Transformer: information encoding

- Normalization

  → Controls the dynamics of the token values before each key step

  - μ, σ: computed over all the tokens corresponding to an image
  - γ, β: parameters to be learned



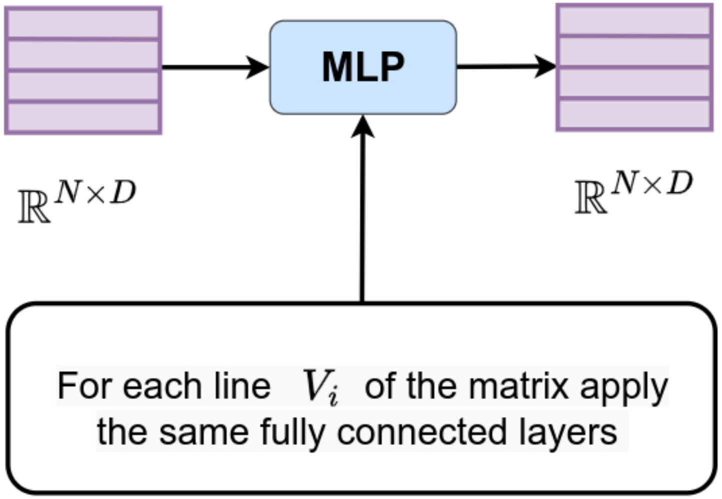
Norm
$\mathbb{R}^{N \times D}$
$\mathbb{R}^{N \times D}$
For each line V_i of the matrix do
$\gamma\left(\frac{V_i - \mu}{\sigma}\right) + \beta$

Transformer: information encoding

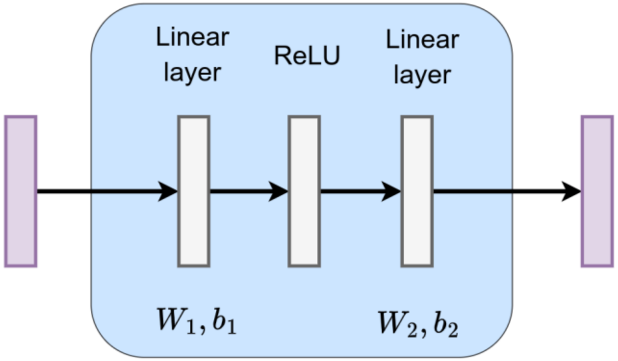
- MLP
  - ➔ Introduces non-linearity
  - ➔ Enables the generation of relevant information

$$z_l{}^* = LN(z_l{}')$$

$$MLP(z_l{}^*) = max(0, z_l{}^* W_1 + b_1) W_2 + b_2$$

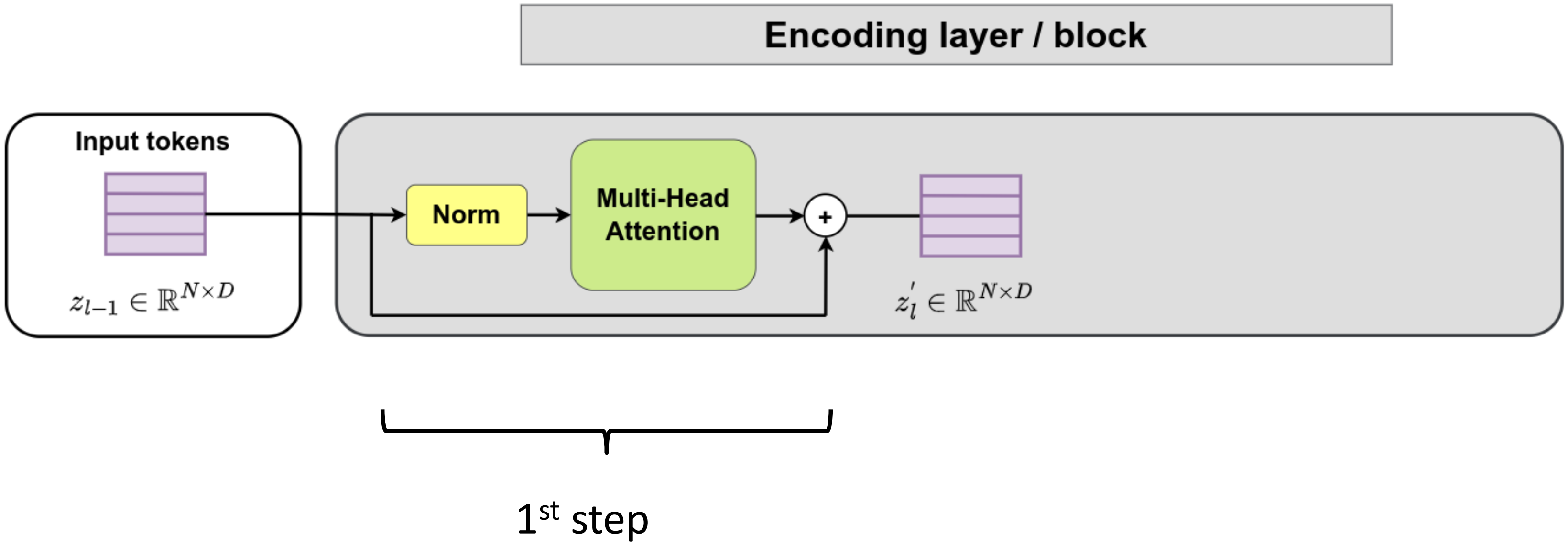


For each line V_i of the matrix apply the same fully connected layers



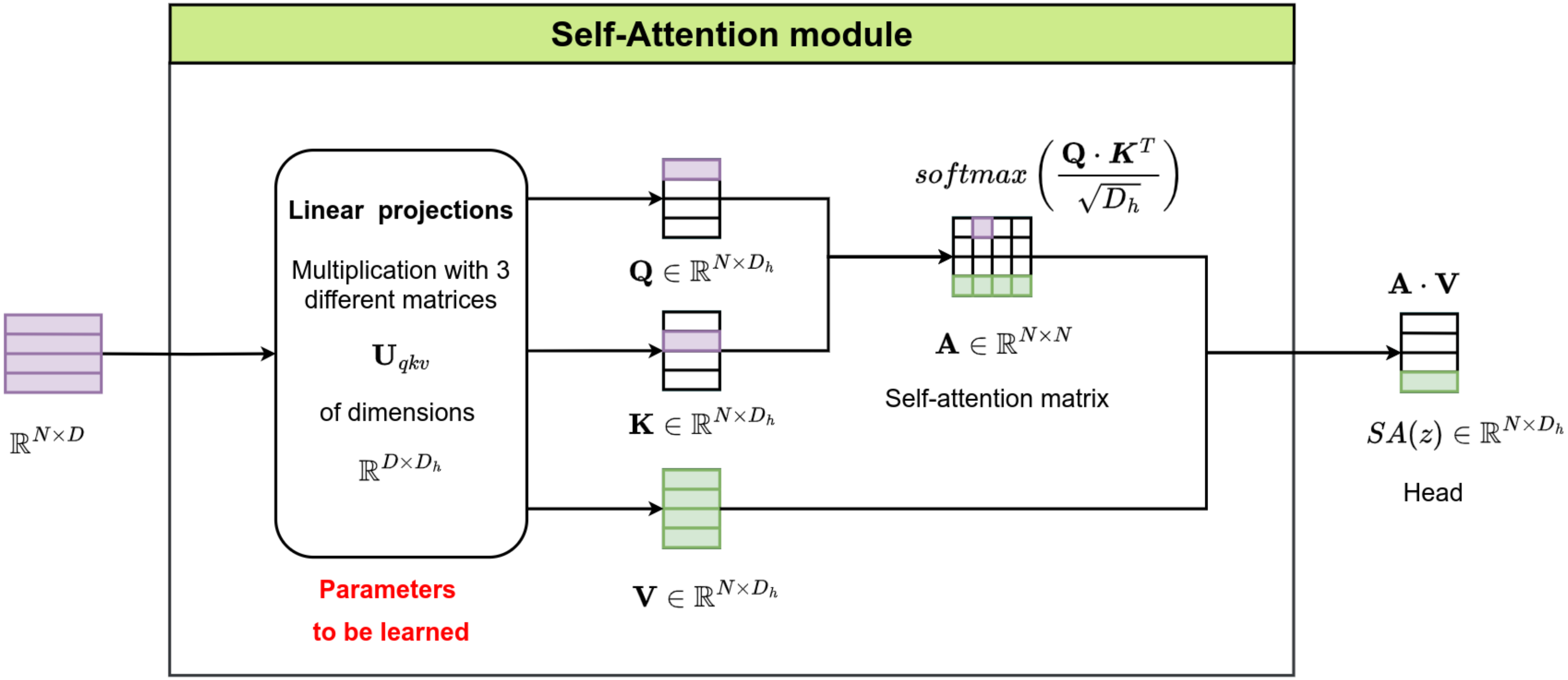
Transformer: information encoding

- Focus on the 1st step


Encoding layer / block
Input tokens
$z_{l-1} \in \mathbb{R}^{N \times D}$
Norm
Multi-Head Attention
+
$z_l^{'} \in \mathbb{R}^{N \times D}$
1st step

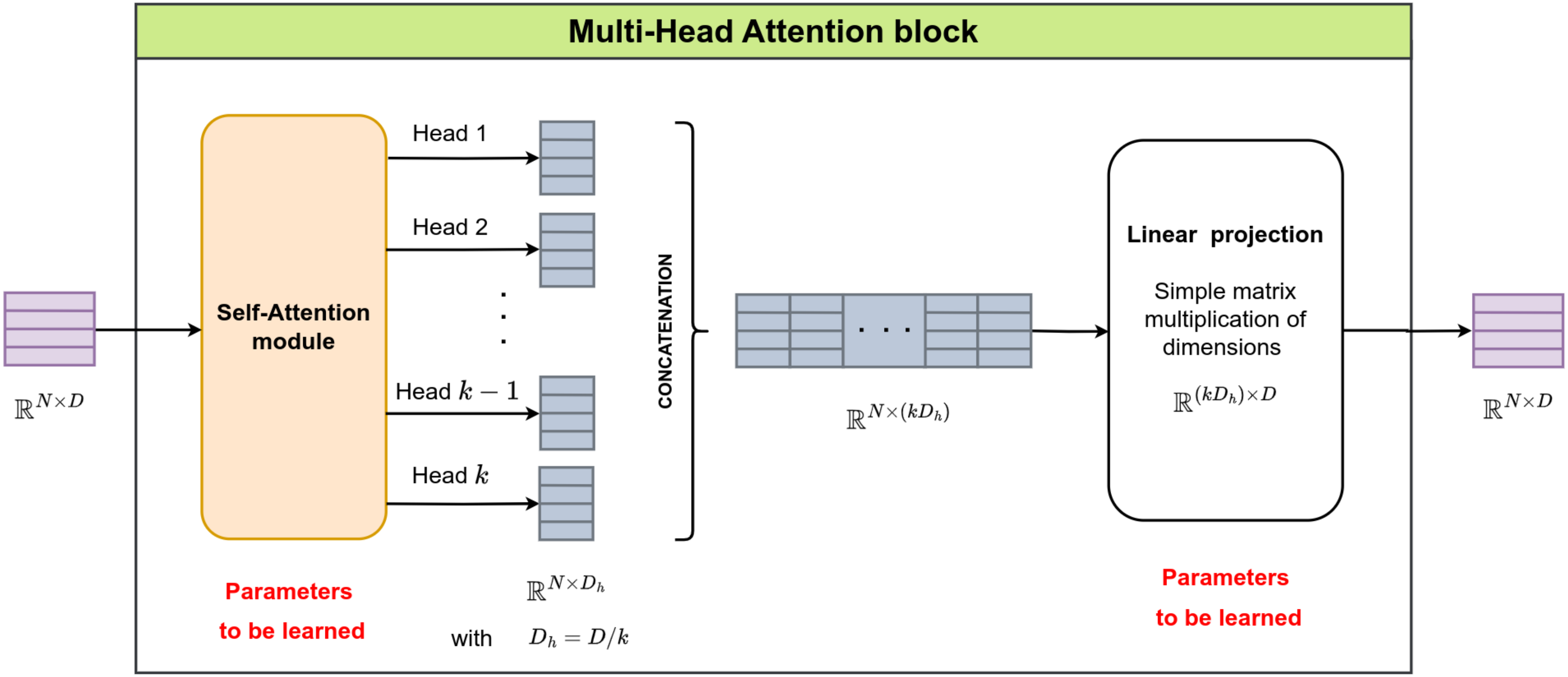
Transformer: information encoding

- Self-attention module
  - ➔ Management of attention maps A through Q (query), K (key), V (values) matrices
  - ➔ Softmax applied row-wise to the matrix A to normalize the weights that will weight the row vectors of V


Self-Attention module
$\mathbb{R}^{N \times D}$
Linear projections
Multiplication with 3 different matrices
$\mathbf{U}_{qkv}$
of dimensions
$\mathbb{R}^{D \times D_h}$
Parameters to be learned
$\mathbf{Q} \in \mathbb{R}^{N \times D_h}$
$\mathbf{K} \in \mathbb{R}^{N \times D_h}$
$\mathbf{V} \in \mathbb{R}^{N \times D_h}$
$softmax\left(\frac{\mathbf{Q} \cdot \boldsymbol{K}^T}{\sqrt{D_h}}\right)$
$\mathbf{A} \in \mathbb{R}^{N \times N}$
Self-attention matrix
$\mathbf{A} \cdot \mathbf{V}$
$SA(z) \in \mathbb{R}^{N \times D_h}$
Head

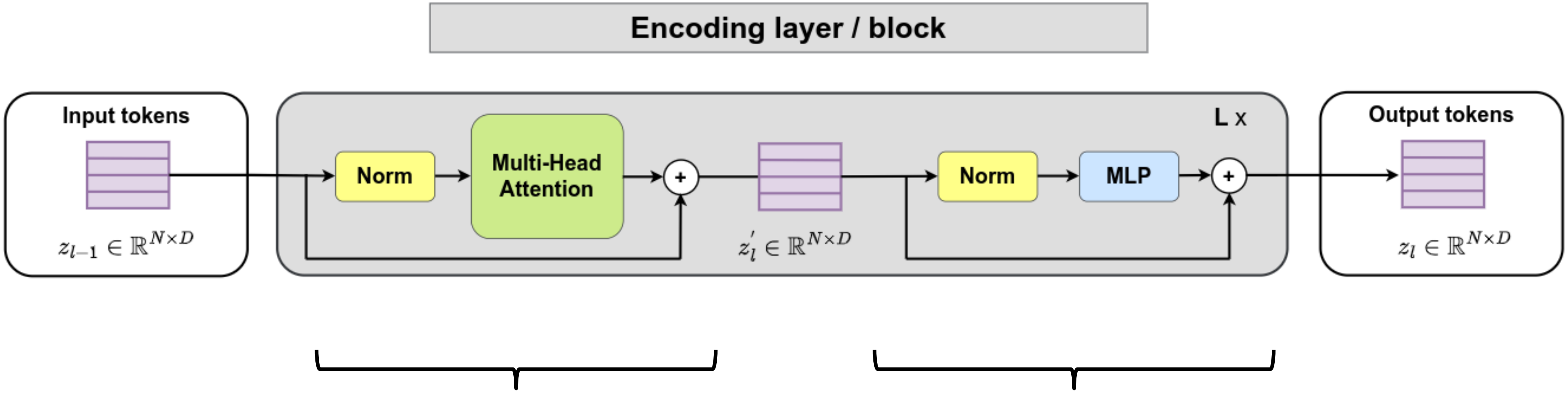
Transformer: information encoding

- Multi-Head Attention: Multi-head attention block
  - ➔ Generation of k heads from different self-attention modules
  - ➔ Equivalent to the concept of feature maps in *CNNs*
  - ➔ Linear projection to mix the information from different heads and return to the initial token dimensions



Transformer: information encoding

- In summary
  - 1st step: generation of information through attention between tokens
  - 2nd step: generation of relevant information through non-linearity


Encoding layer / block
Input tokens
$z_{l-1} \in \mathbb{R}^{N \times D}$
L x
Norm
Multi-Head Attention
+
$z_l' \in \mathbb{R}^{N \times D}$
Norm
MLP
+
Output tokens
$z_l \in \mathbb{R}^{N \times D}$

1st step

2nd step

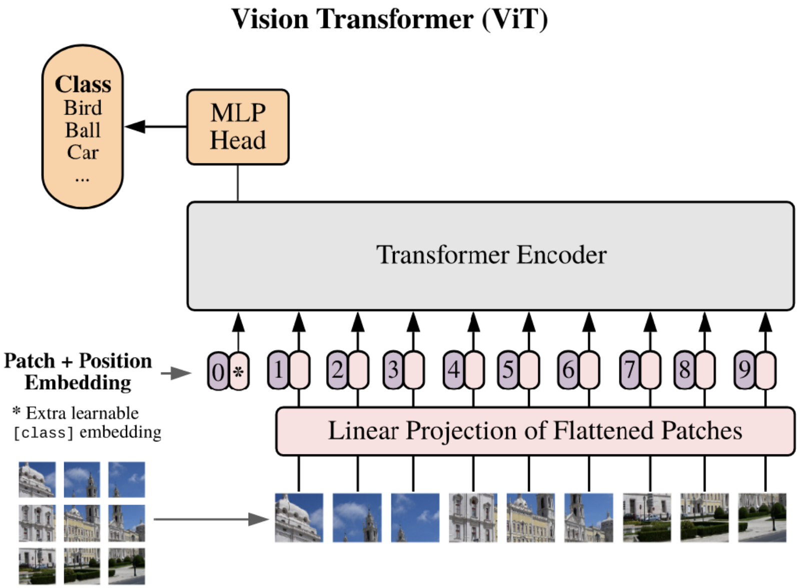
Transformers

Classification method

Transformer for classification

- VIT: reference algorithm
  - → Trained on JFT (300 million images)
  - → Introduction of the concept of *class token*

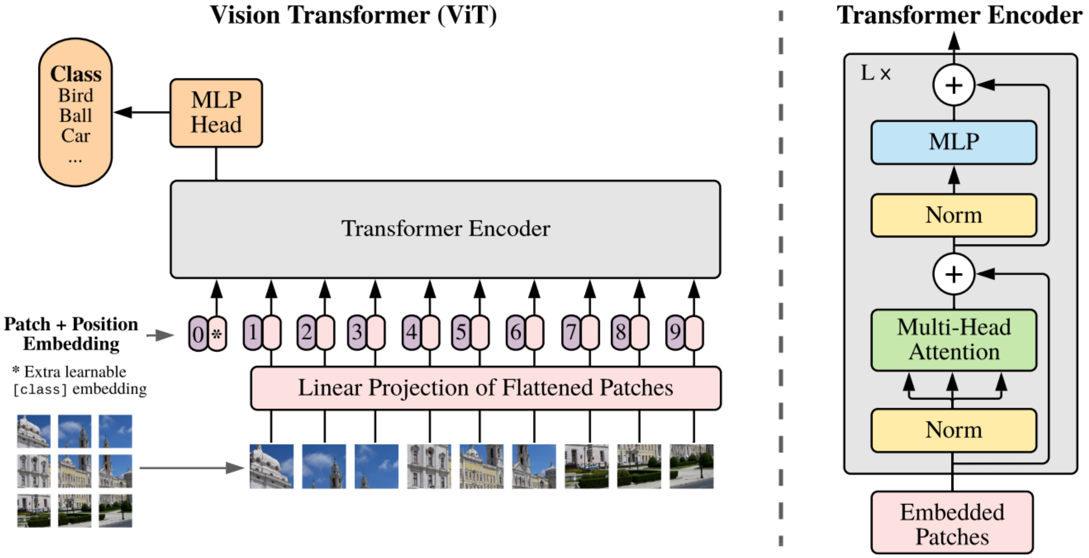
Learning a “pooling” operation with respect to visual tokens


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

gif animation

Transformer for classification

- VIT: reference algorithm



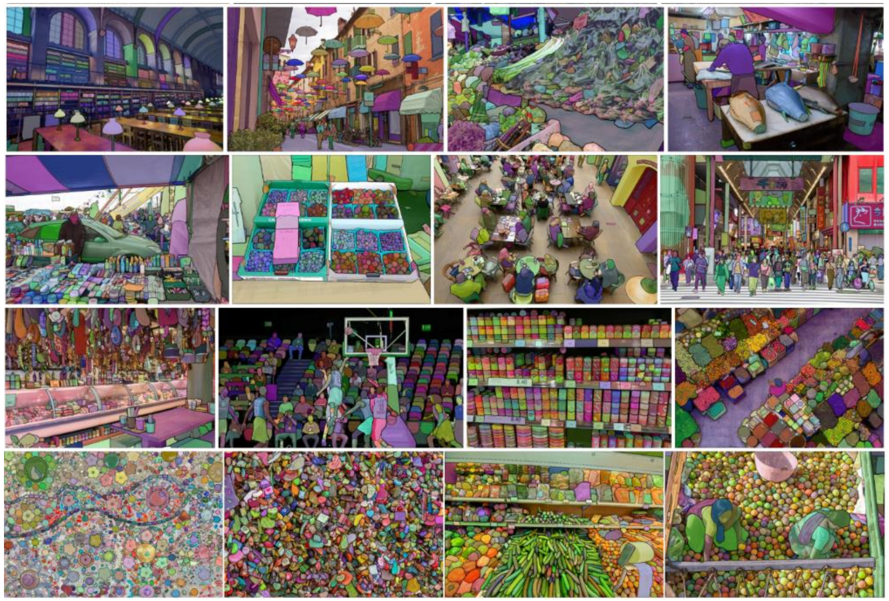
Models	Layers / blocks	Hidden size D	MLP size	Heads	Parameters
ViT-Base	12	768	3072	12	86 M
ViT-Large	24	1024	4096	16	307 M
ViT-Huge	32	1280	5120	16	632 M

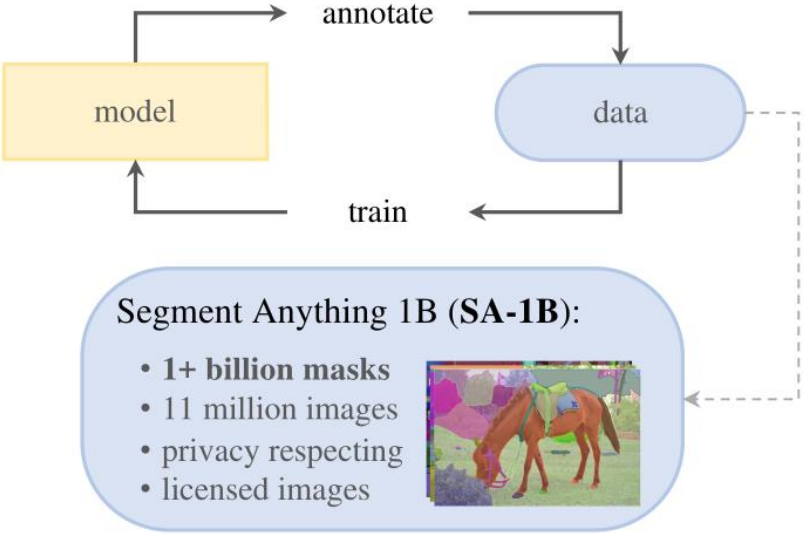
Transformers

Segmentation model

Transformer for segmentation

- SAM: Segment Anything
  - Trained on SA-1B (11 million images, 1 billion masks)
  - 2D images of natural scenes, Minimum side size: 1500px



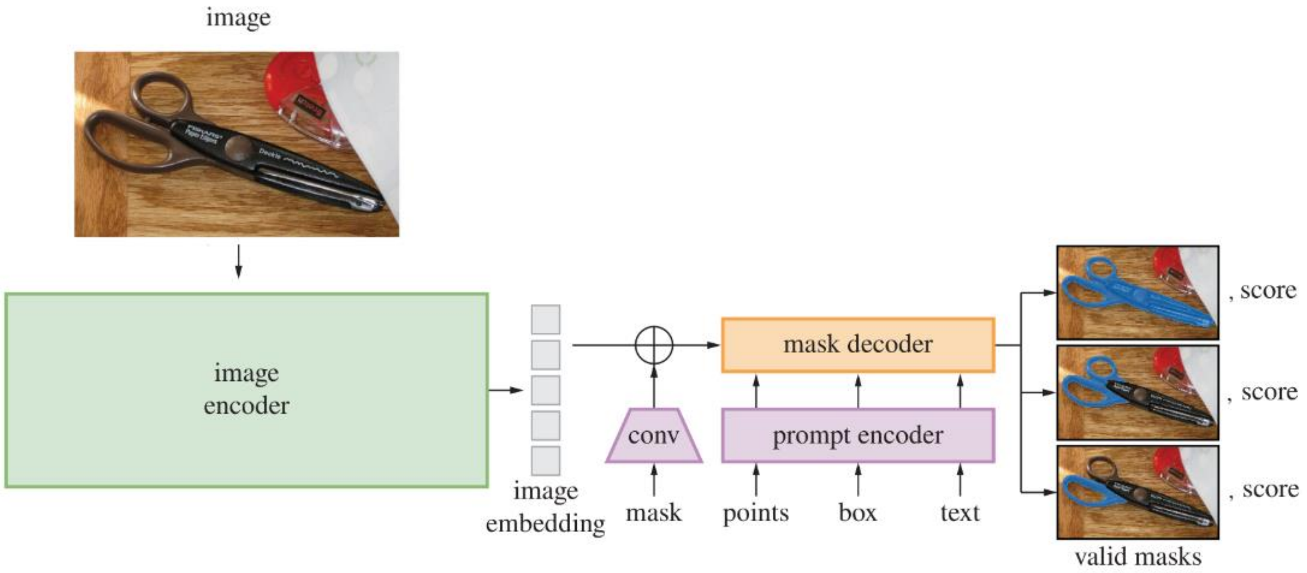


Segment Anything 1B (**SA-1B**):

- **1+ billion masks**
- 11 million images
- privacy respecting
- licensed images

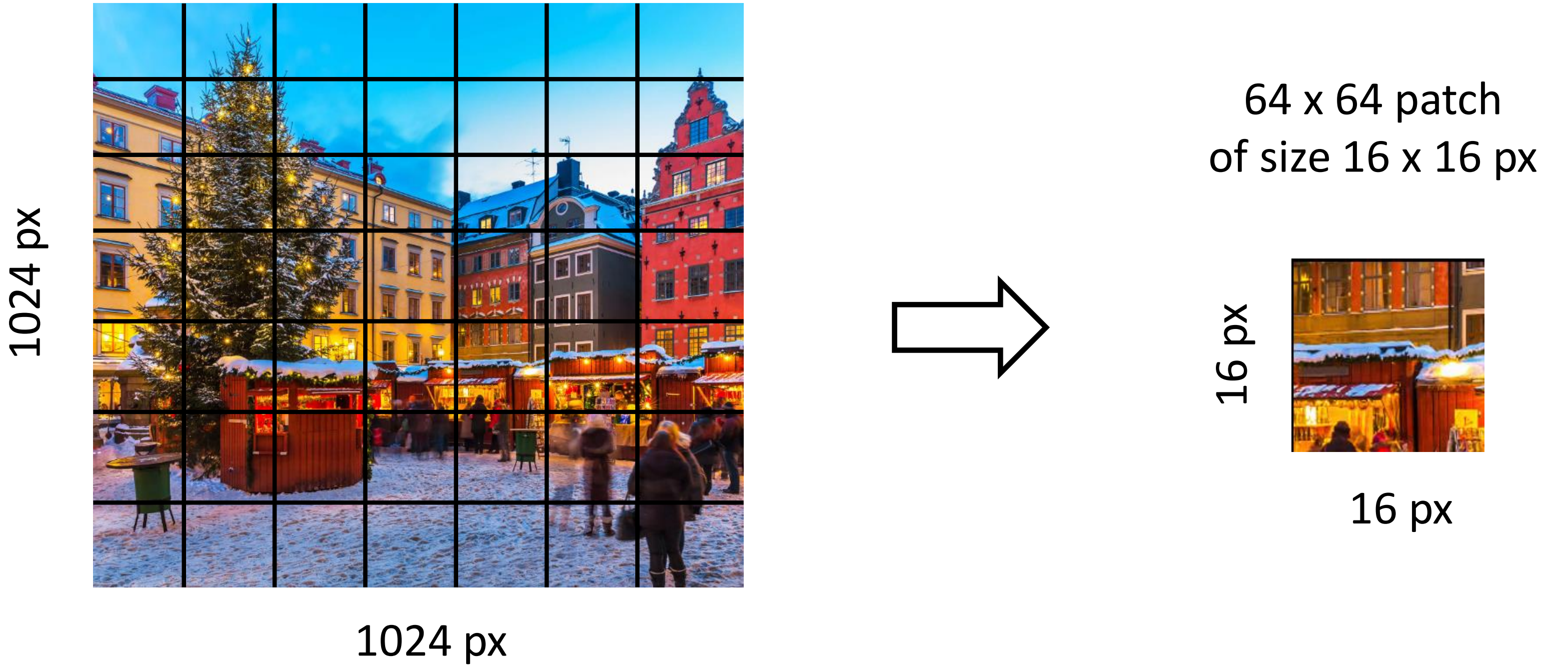
Transformer for segmentation

- SAM: Segment Anything
  - ➔ Relatively simple architecture
  - ➔ Interactive segmentation process
  - ➔ Integrates the principle of segmentation ambiguity based on initialization



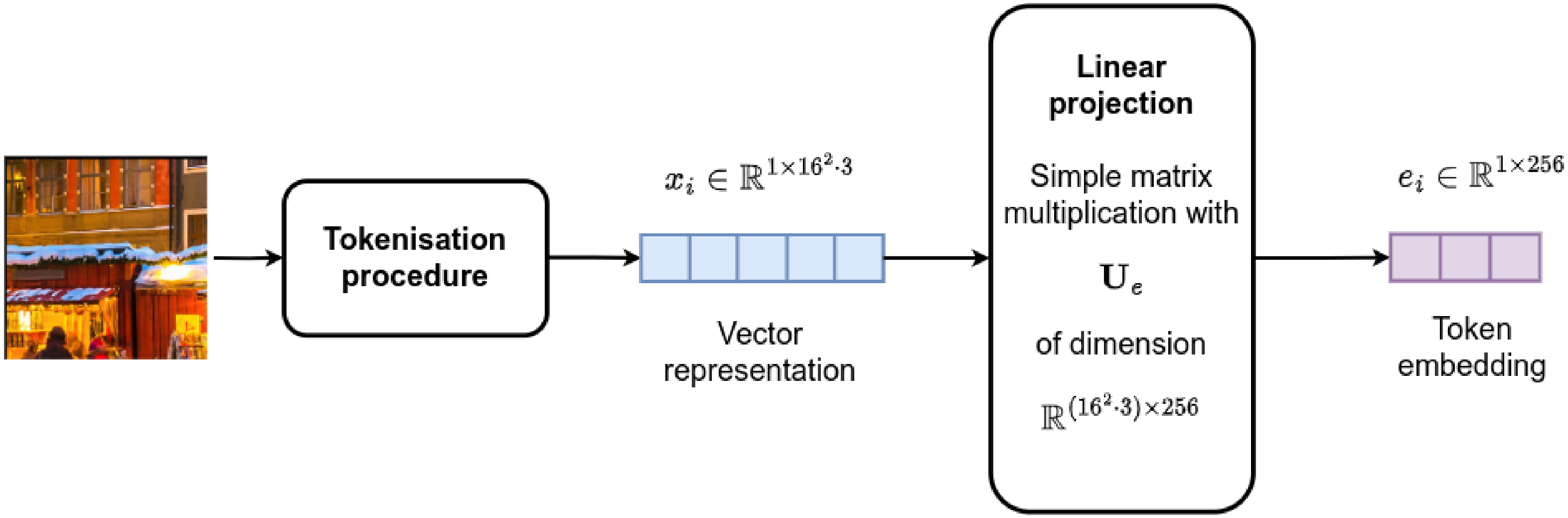
Transformer for segmentation

- SAM: Segment Anything
  - ➔ Image encoder
    - Input images resized to 1024 x 1014 px
    - Architecture: ViT-H/16 (token = patch of size 16 x 16 px)
    - Hidden size D: 256



Transformer for segmentation

- SAM: Segment Anything
  - ➔ Image encoder
    - Input images resized to 1024 x 1014 px
    - Architecture: ViT-H/16 (token = patch of size 16 x 16 px)
    - Hidden size D: 256



Tokenisation procedure

$x_i \in \mathbb{R}^{1\times 16^2\cdot 3}$

Vector representation

Linear projection

Simple matrix multiplication with

$\mathbf{U}_e$

of dimension

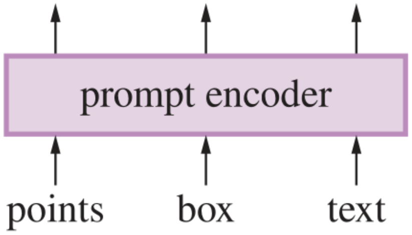
$\mathbb{R}^{(16^2\cdot 3)\times 256}$

$e_i \in \mathbb{R}^{1\times 256}$

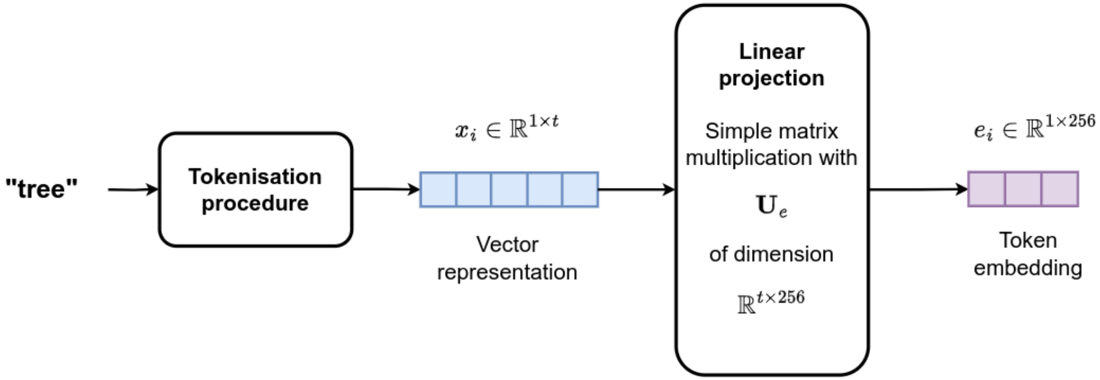
Token embedding

Transformer for segmentation

- SAM: Segment Anything
  - ➔ Prompt encoder
    - Several possible prompts: points, bounding boxes, text
    - Hidden size D: 256

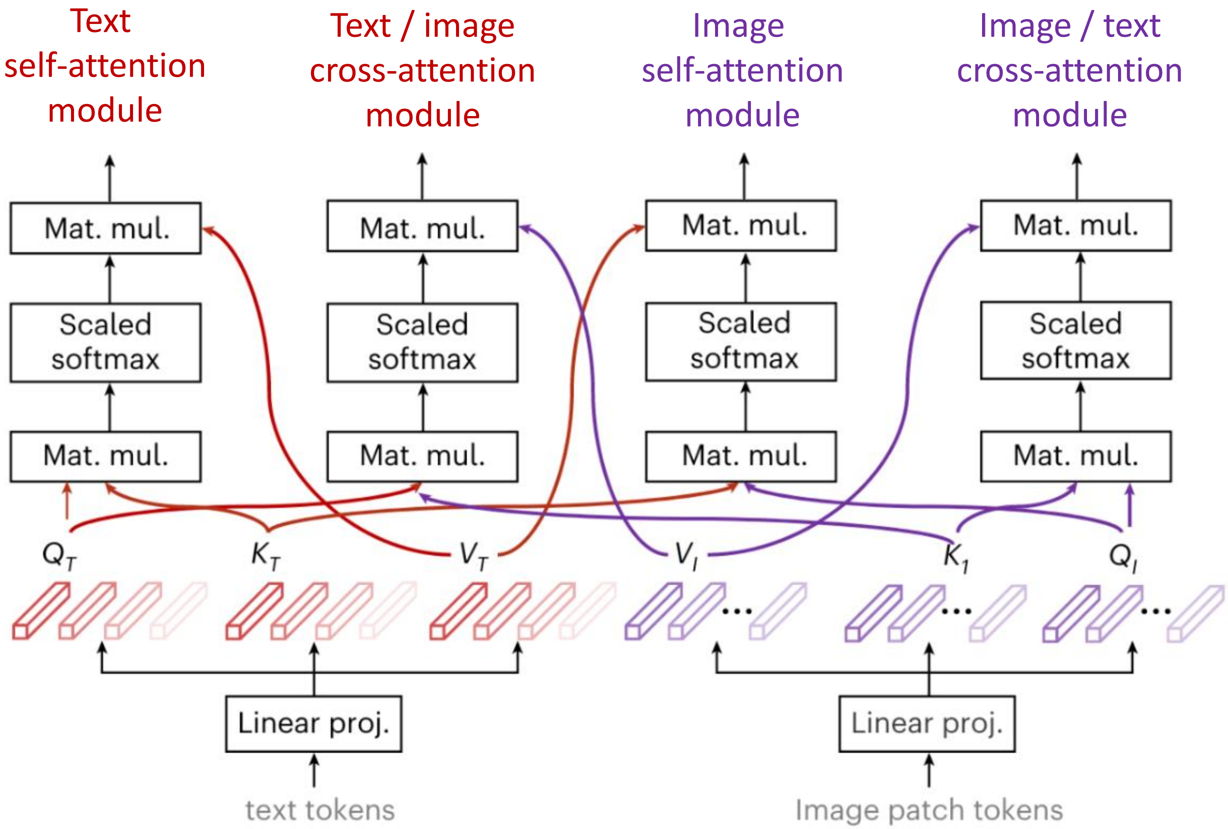


- Illustration for the text encoder


"tree"
Tokenisation procedure
$x_i \in \mathbb{R}^{1 \times t}$
Vector representation
Linear projection
Simple matrix multiplication with
$\mathbf{U}_e$
of dimension
$\mathbb{R}^{t \times 256}$
$e_i \in \mathbb{R}^{1 \times 256}$
Token embedding

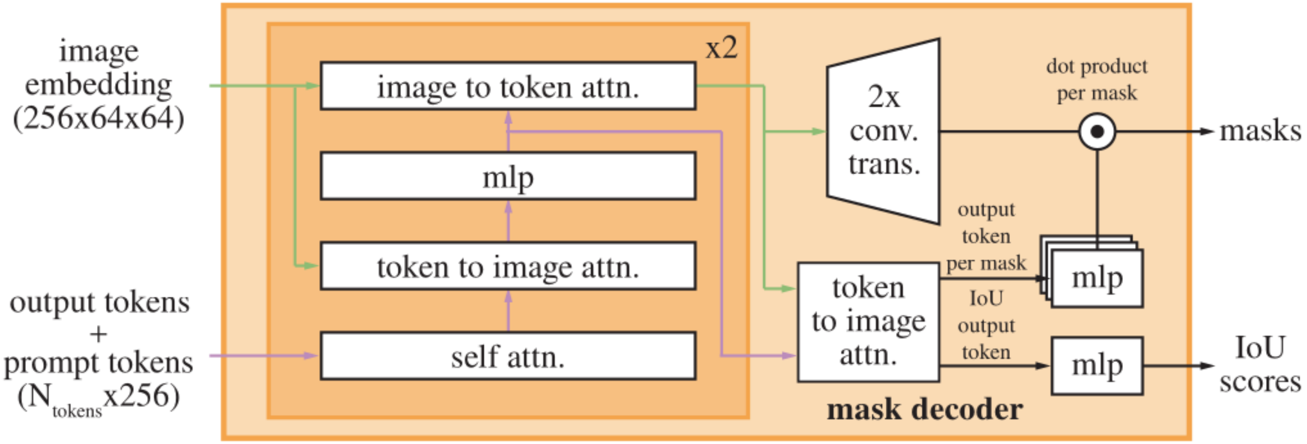
Transformer for segmentation

- SAM: Segment Anything
  - ➔ Mask decoder
    - Uses the cross-attention principle



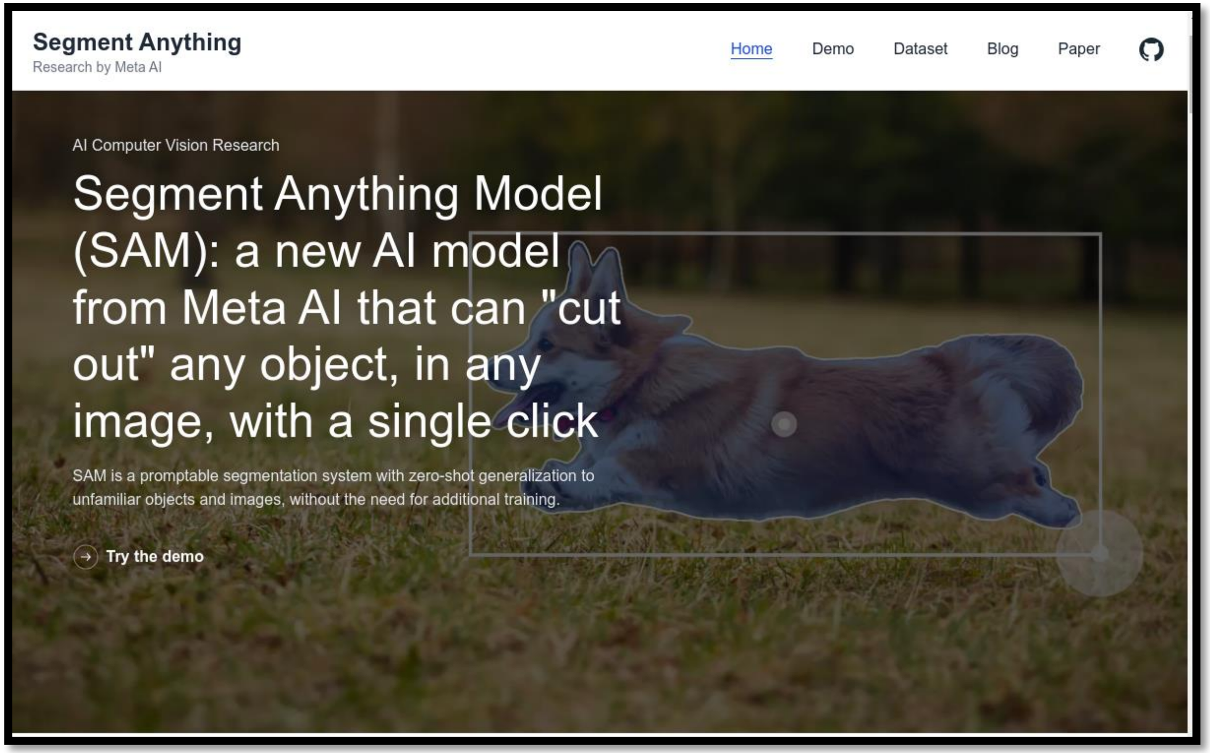
Transformer for segmentation

- SAM: Segment Anything
  - ➔ Mask decoder
    - Integration of a class token (output token)



Transformer for segmentation

- SAM: Segment Anything
  - Link to demo
  - https://segment-anything.com/



AI Computer Vision Research

Segment Anything Model (SAM): a new AI model from Meta AI that can "cut out" any object, in any image, with a single click

SAM is a promptable segmentation system with zero-shot generalization to unfamiliar objects and images, without the need for additional training.

That’s all folks