Attention and Transformers

Lyon, Avril 21- 25

DLMI2025

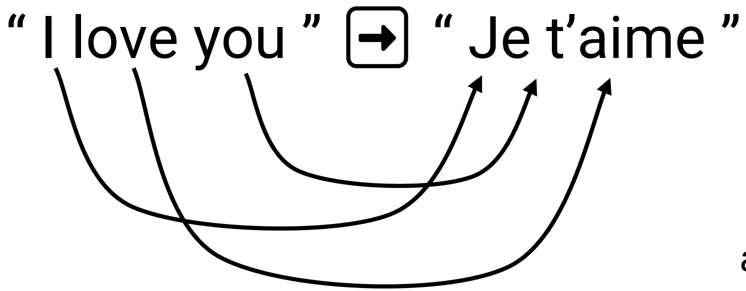
Deep Learning for Medical Imaging, Lyon

with the collaboration of Olivier Bernard

What is Attention?

- **Definition:** Relative importance of elements in a sequence
  - Mechanism to prioritize parts of the input
- Introduced for *Natural Language Processing* (NLP) in 2014/15
  - Alternative to encoder-decoder for sequence-to-sequence tasks

Translate " I love you " in French:

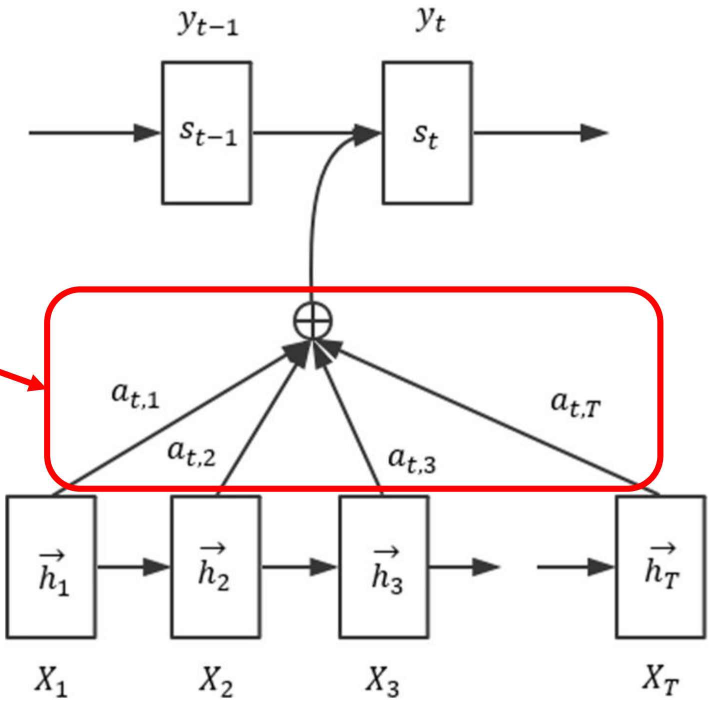


	I	love	you
je	0.94	0.02	0.04
t'	0.11	0.01	0.88
aime	0.03	0.95	0.02

What Are Transformers?

Attention (introduced by Bahdanau *et al.*, ICLR 2015):

- Attention weights in *Recurrent Neural Networks* (RNN)



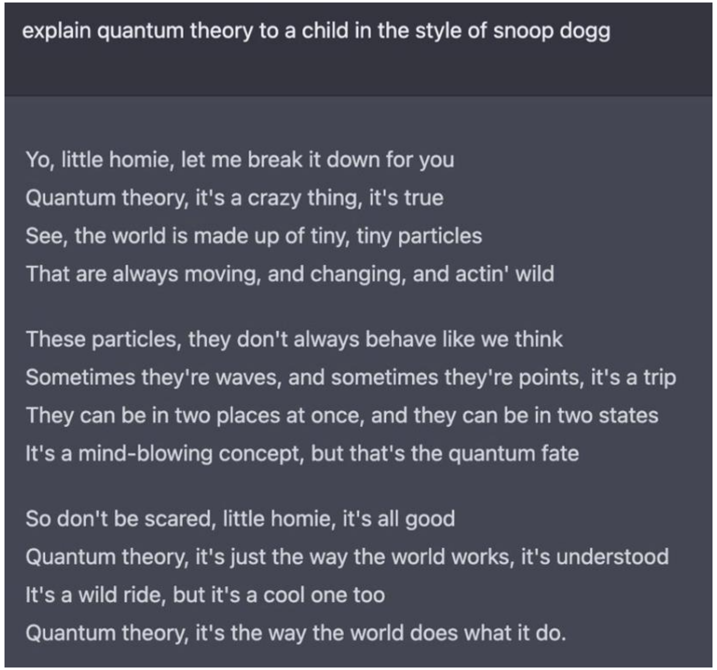
Transformers (Vaswani *et al.*, NeurIPS 2017):
"Attention Is All You Need" !

- Pipeline built around attention

Applications to Natural Language Processing

- Tasks formulated for Transformers
  - Machine translation
  - Speech comprehension
  - Text generation
  - Computer code generation

- Popularized by models like
  - BERT
  - GPT 3/4
  - Chat-GPT
  - and many more…



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

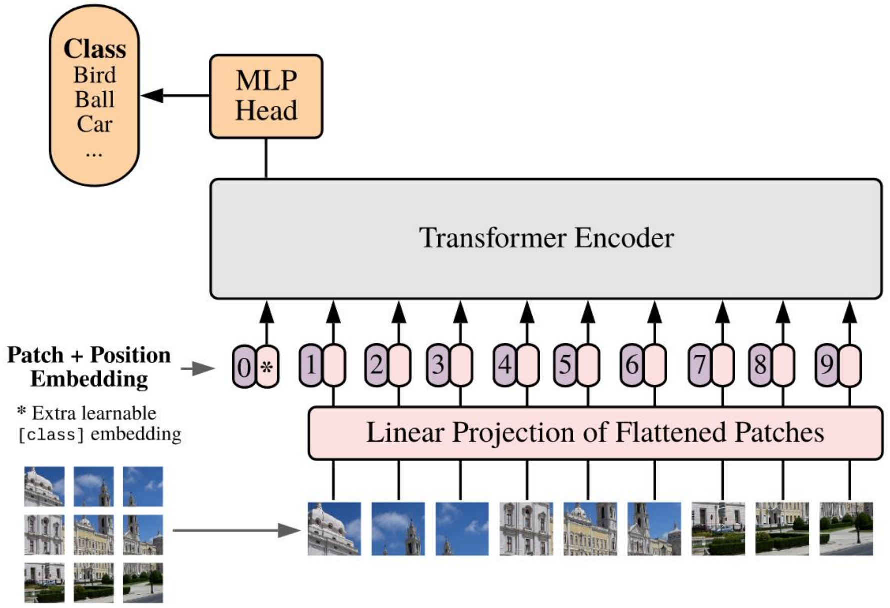
So don't be scared, little homie, it's all good
Quantum theory, it's just the way the world works, it's understood
It's a wild ride, but it's a cool one too
Quantum theory, it's the way the world does what it do.

Generalization to Images (and other domains)

Seminal architectures

- Classification
- Segmentation
- Segmentation + time
- Image generation

Vision Transformer (ViT) - 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

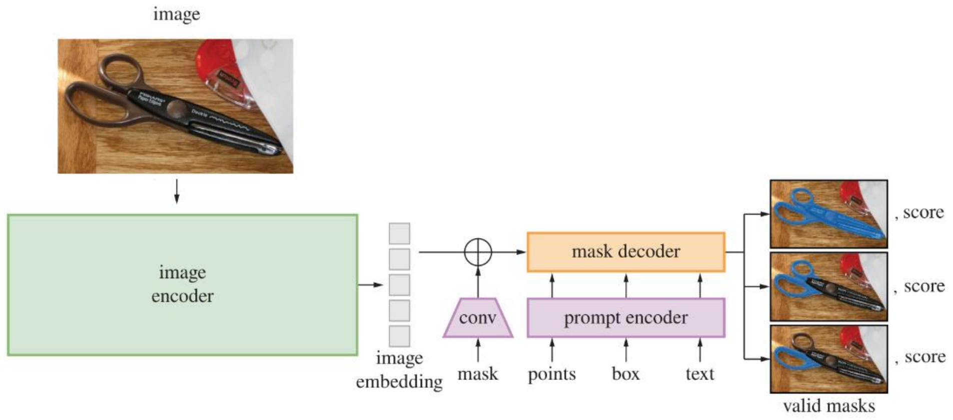
[Dosovitskiy *et al.*, ICLR 2021]

Generalization to Images (and other domains)

Seminal architectures

- Classification
- Segmentation
- Segmentation + time
- Image generation

Segment Anything Model (SAM) - 2023



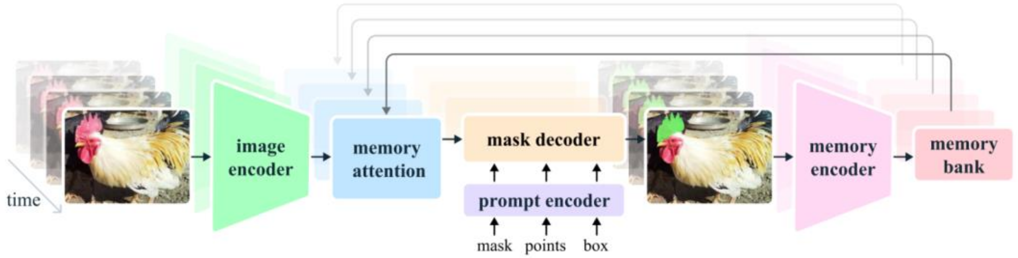
[Kirillov *et al.*, ICCV 2023]

Generalization to Images (and other domains)

Seminal architectures

- Classification
- Segmentation
- Segmentation + time
- Image generation

SAM 2 - 2024



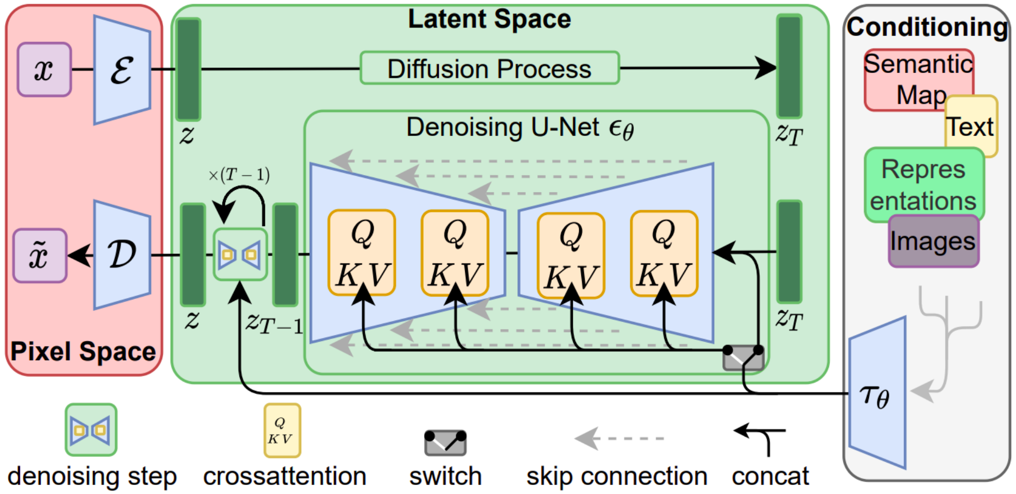
[Ravi *et al.*, arXiv 2024]

Generalization to Images (and other domains)

Seminal architectures

- Classification
- Segmentation
- Segmentation + time
- Image generation

Latent Diffusion - 2022



[Rombach *et al.*, CVPR 2022]

Transformer Architecture

How to Represent Data?

Break down input into *tokens*, i.e. vectors

- Text: token = word of a sentence
- Images: token = patch of an image

Text

"I love you"

Tokenization

"I" "love" "you"

"I" $\Rightarrow x_i \in \mathbb{R}^t$

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

Image

Tokenization

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

with P patch size,

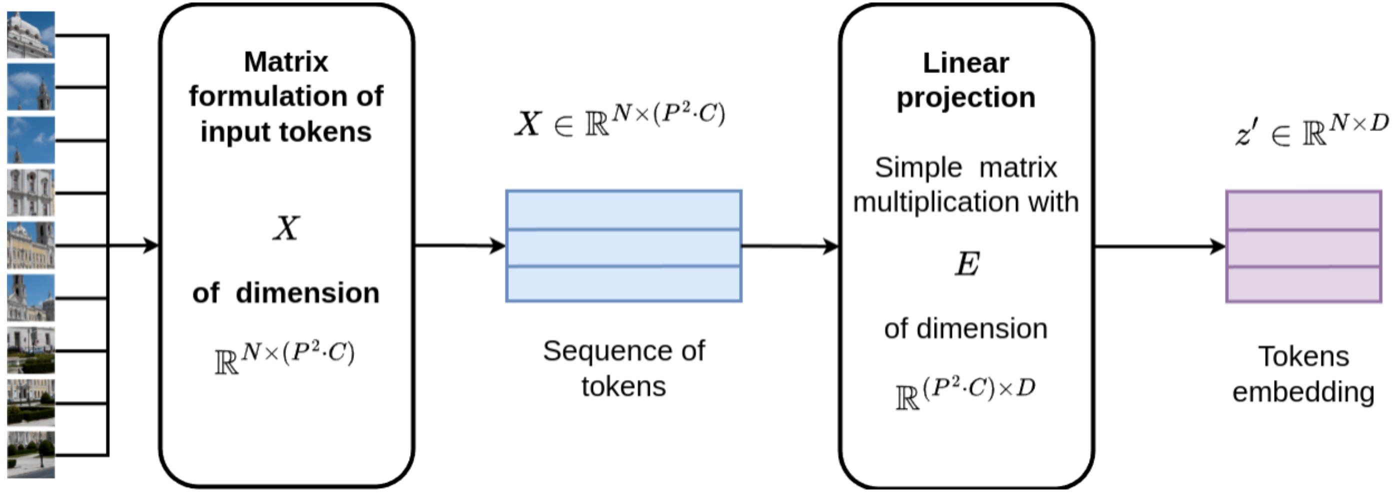
C number of channels

How to Represent Data?

Represent token in a common space, i.e. embedding

- Linear projection: Multiply by a learnable matrix

$$\boxed{z' = XE}$$


Matrix formulation of input tokens
X
of dimension
$\mathbb{R}^{N \times (P^2 \cdot C)}$
$X \in \mathbb{R}^{N \times (P^2 \cdot C)}$
Sequence of tokens
Linear projection
Simple matrix multiplication with
E
of dimension
$\mathbb{R}^{(P^2 \cdot C) \times D}$
$z' \in \mathbb{R}^{N \times D}$
Tokens embedding

How to Represent Data?

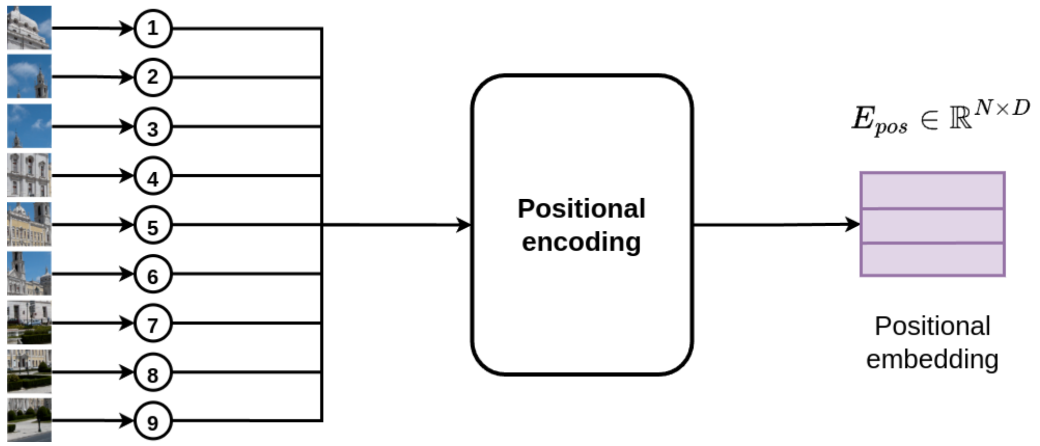
Transformers process tokens as unordered set

✓ Fast parallelizable attention…

! Loss of structural information ⇒ Permutation invariance!

To recover structure:

- *Positional encoding* (PE)
    - Add positional info to each token



How to Represent Data?

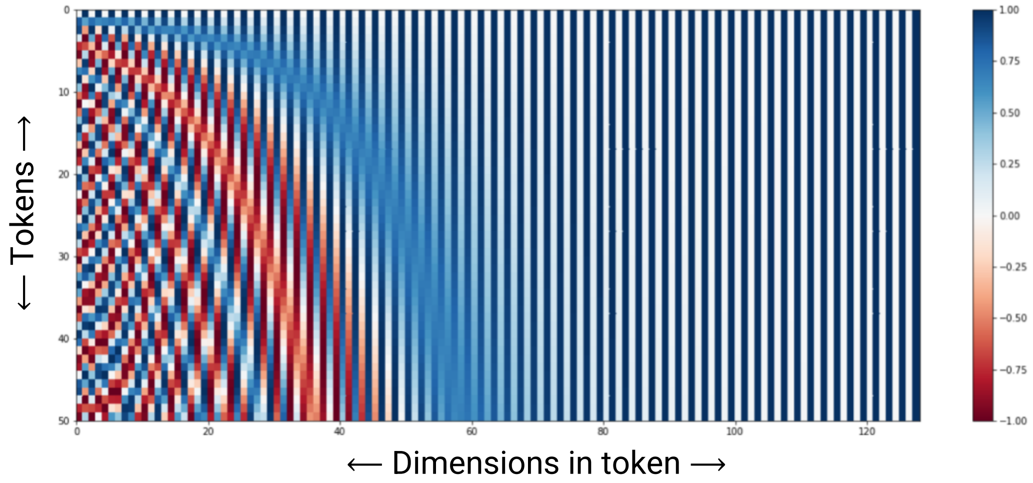
Positional encoding (PE)

[Reference: HuggingFace blog]

- Typical choice: sinusoidal function

$$E_{\text{pos}}(n, 2d) = \sin\left(\frac{n}{10\,000^{2d/D}}\right)$$

$$E_{\text{pos}}(n, 2d+1) = \cos\left(\frac{p}{10\,000^{2d/D}}\right)$$

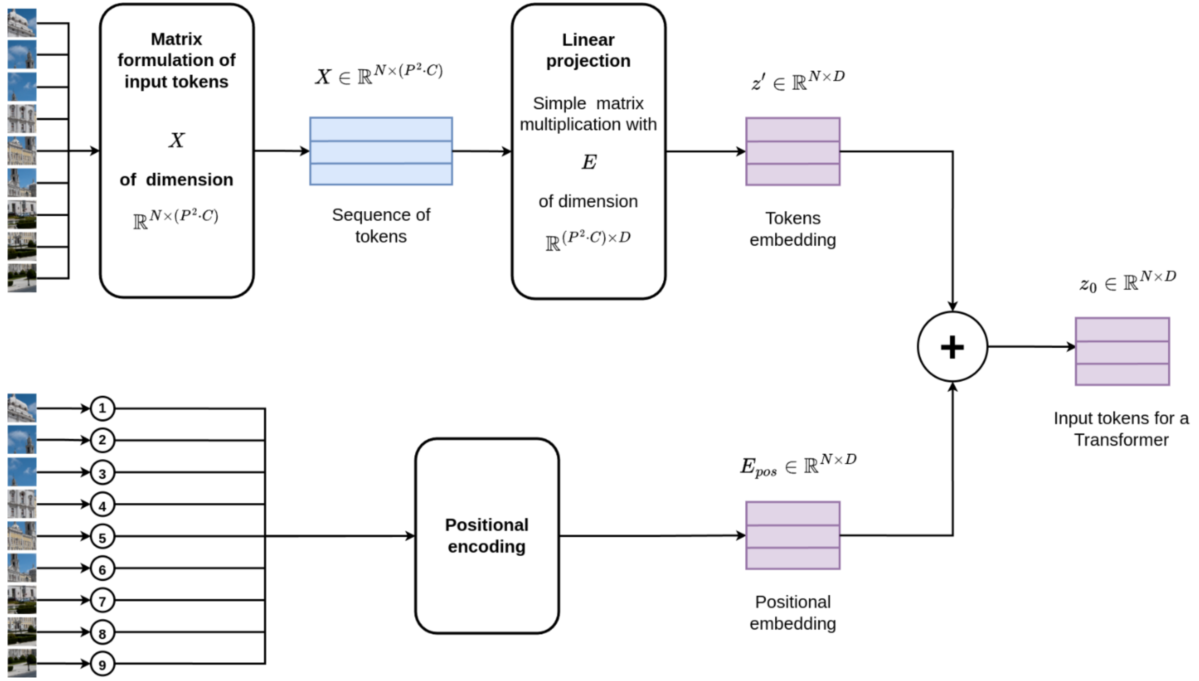


- Other option: learnable parameters $E_{\text{pos}} \in \mathbb{R}^{N \times D} \sim \mathcal{N}(0, 0.02)$

How to Represent Data?

Putting the pieces together

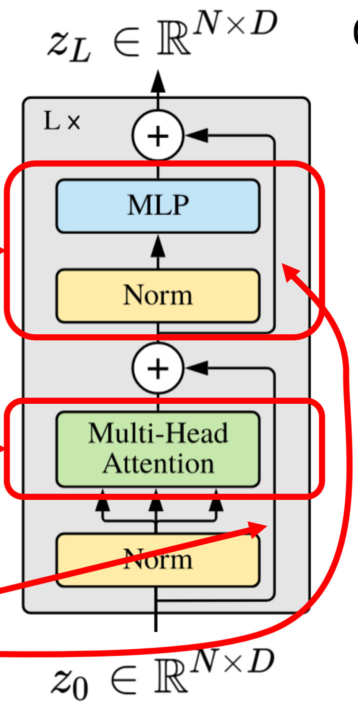
- Final tokens = embeddings + position encoding
- Parameters:
  - Projection $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$
  - PE (optional) $E_{\text{pos}} \in \mathbb{R}^{N \times D}$


Matrix formulation of input tokens
X
of dimension
$\mathbb{R}^{N\times(P^2\cdot C)}$
$X \in \mathbb{R}^{N\times(P^2\cdot C)}$
Sequence of tokens
Linear projection
Simple matrix multiplication with
E
of dimension
$\mathbb{R}^{(P^2\cdot C)\times D}$
$z' \in \mathbb{R}^{N\times D}$
Tokens 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

How to Process the Tokens?

Transformer block:

- Intra-token computation
  - MLP, normalization
- Inter-token communication
  - Attention
- Residual connections
  - Propagate gradients + PE


$z_L \in \mathbb{R}^{N \times D}$
L x
MLP
Norm
Multi-Head Attention
Norm
$z_0 \in \mathbb{R}^{N \times D}$

Output tokens

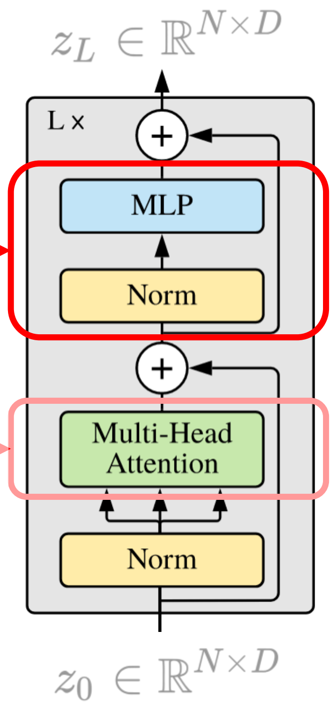
[Dosovitskiy *et al.*, ICLR 2021]

Input tokens

Transformer Block - Computation

Transformer block:

- Intra-token computation
  - MLP, normalization
- Inter-token communication
  - Attention
- Residual connections
  - Propagate gradients + PE


$z_L \in \mathbb{R}^{N \times D}$
L x
+
MLP
Norm
+
Multi-Head Attention
Norm
$z_0 \in \mathbb{R}^{N \times D}$

Output tokens

Input tokens

[Dosovitskiy *et al.*, ICLR 2021]

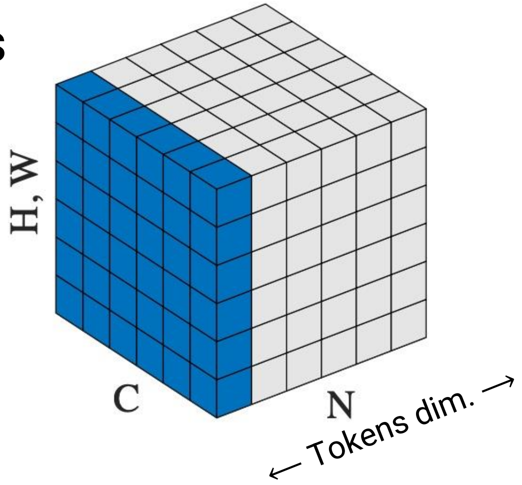
Transformer Block - Computation

Intra-token computation layers operate on each token separately

- Layer Normalization (LN)
  - μ, σ : computed **for each token, i.e. image**
  - γ, β : learnable affine transform. parameters

$$\tilde{z}_{l,i} = \gamma\left(\frac{z_{l,i}-\mu}{\sigma}\right)+\beta$$

with $z_{l,i}$ a token, i.e. row, in tokens z_l



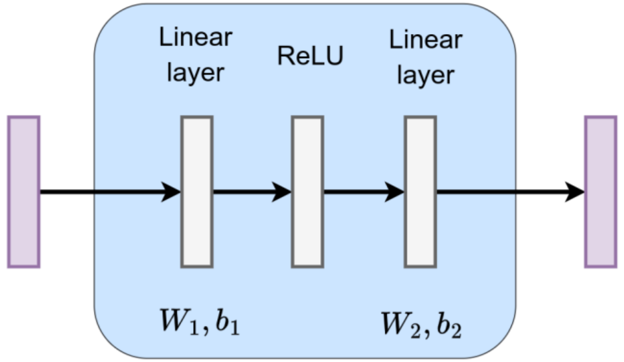
[Source: PyTorch Documentation]

Transformer Block - Computation

Intra-token computation layers operate on each token separately

- Feed-Forward Network (MLP)
    - Add non-linearity
    - Refine each token's representation

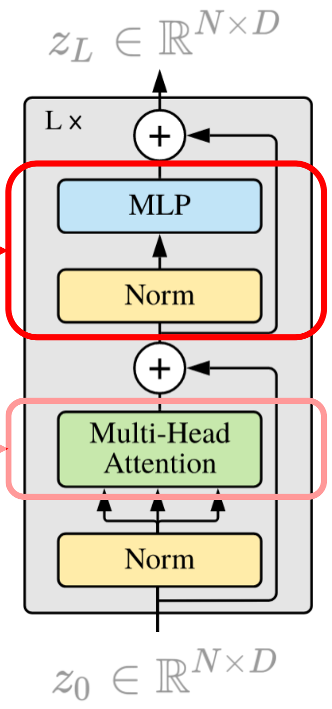
$$\begin{aligned} \tilde{z}_{l,i} &= \mathrm{LN}(z_{l,i}) \\ \mathrm{MLP}(\tilde{z}_{l,i}) &= \max(0, \tilde{z}_{l,i} W_1 + b_1) W_2 + b_2 \end{aligned}$$



Transformer Block - Computation

Transformer block:

- Intra-token computation
  - MLP, normalization
- Inter-token communication
  - Attention
- Residual connections
  - Propagate gradients + PE



$z_L \in \mathbb{R}^{N \times D}$
L x
MLP
Norm
Multi-Head Attention
Norm
$z_0 \in \mathbb{R}^{N \times D}$

Output tokens

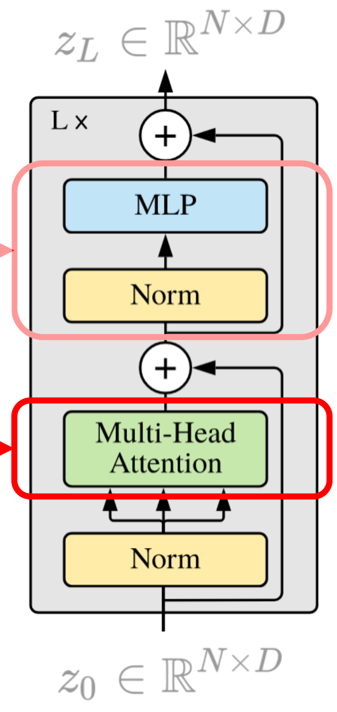
Input tokens

[Dosovitskiy *et al.*, ICLR 2021]

Transformer Block - Communication

Transformer block:

- Intra-token computation
  - MLP, normalization
- Inter-token communication
  - Attention
- Residual connections
  - Propagate gradients + PE


$z_L \in \mathbb{R}^{N \times D}$
L x
+
MLP
Norm
+
Multi-Head Attention
Norm
$z_0 \in \mathbb{R}^{N \times D}$

Output tokens

Input tokens

[Dosovitskiy *et al.*, ICLR 2021]

Transformer Block - Communication

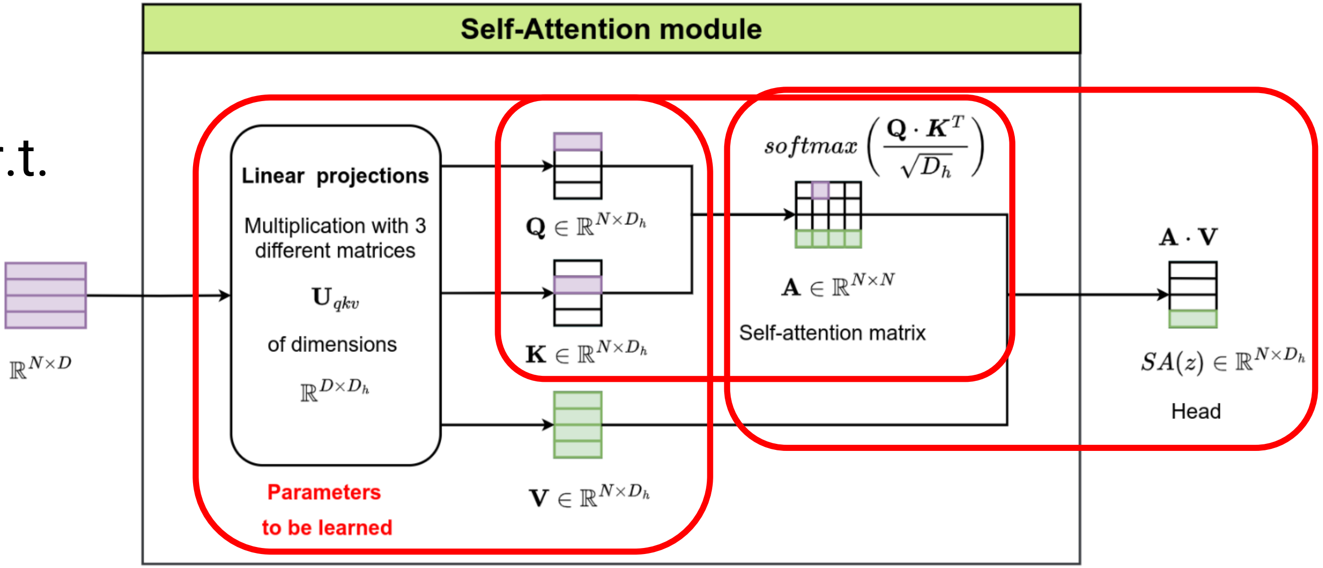
Attention layers exchange information between tokens

- Self-attention

Define attention *A* w.r.t.

- *Query* (**Q**)
- *Key* (**K**)
- *Value* (**V**)

matrices

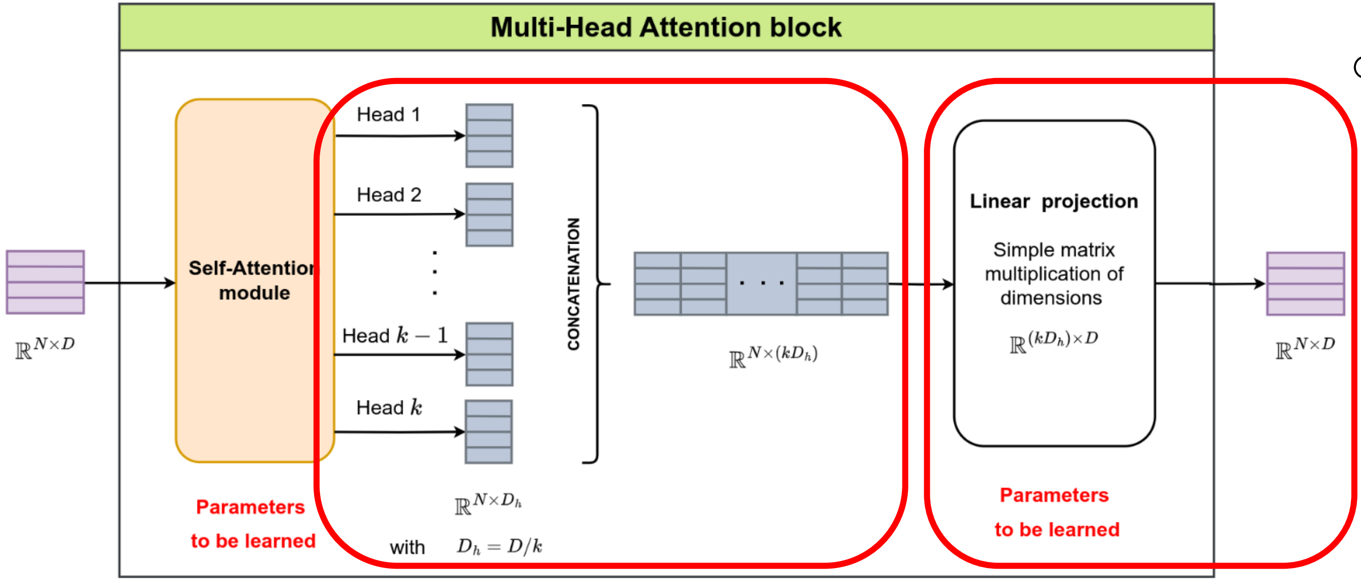

Self-Attention module
$\mathbb{R}^{N \times D}$
Linear projections
Multiplication with 3 different matrices
$\mathbf{U}_{qkv}$
of dimensions
$\mathbb{R}^{D \times D_h}$
Parameters to be learned
$\mathbf{Q} \in \mathbb{R}^{N \times D_h}$
$\mathbf{K} \in \mathbb{R}^{N \times D_h}$
$\mathbf{V} \in \mathbb{R}^{N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

Matrix mul. **Q** **K** to obtain *attention map* **A**

Row-wise softmax on **A** to normalize weights applied to **V**

Transformer Block - Communication

- Multi-Head Attention (MHA)
  - *k* heads running parallel self-attention
  - Similar to feature maps in *Convolutional Neural Networks* (CNN)



- Linear projection to mix the heads

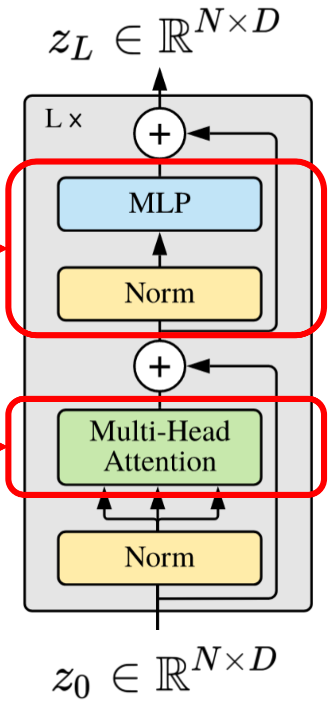
  + return to *D* dimensions (if necessary)

Transformer - Summary

Transformer encoder: transformer blocks repeated *L* times

Transformer block:

- Intra-token computation
  - MLP, normalization
- Inter-token communication
  - Attention
- Residual connections
  - Propagate gradients + PE


$z_L \in \mathbb{R}^{N \times D}$
L ×
MLP
Norm
Multi-Head Attention
Norm
$z_0 \in \mathbb{R}^{N \times D}$

Output tokens

Input tokens

[Dosovitskiy *et al.*, ICLR 2021]

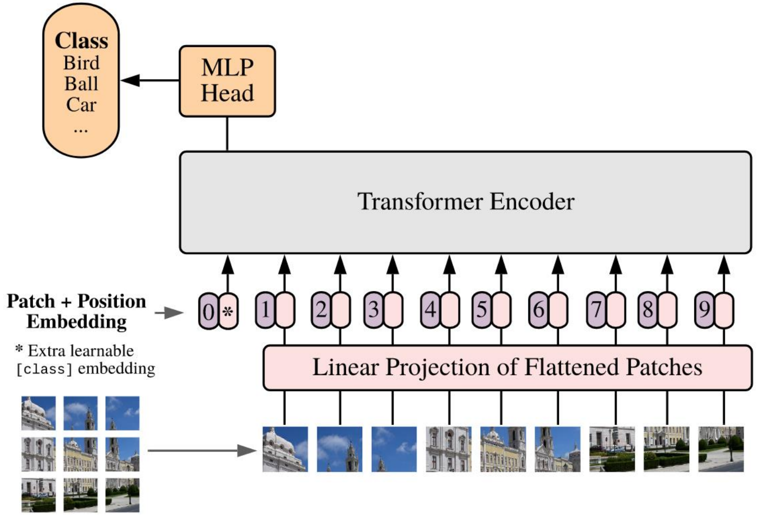
Transformers in Practice

Transformers for Image Classification

Vision Transformer (ViT)

- Reuse *class token* ([CLS]) from BERT

[CLS]: "Pool" information from full sequence of tokens



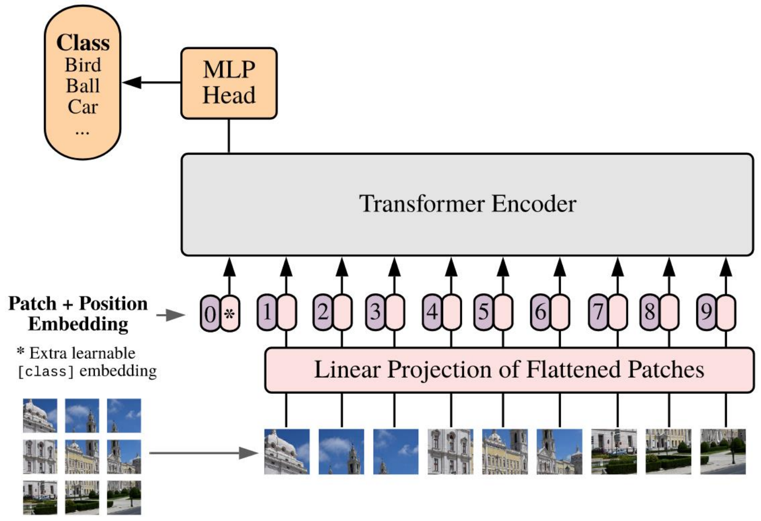
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

[Dosovitskiy *et al.*, ICLR 2021]

Transformers for Image Classification

Vision Transformer (ViT)

- Trained on JFT (300 million images)


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

[Dosovitskiy *et al.*, ICLR 2021]

Models	ViT-Base	ViT-Large	ViT-Huge
Parameters	86 M	307 M	632 M
Layers / blocks	12	24	32
Num. heads	12	16	16
Tokens dim.	768	1024	1280
MLP hidden dim.	3072	4096	5120

Why Use Transformers?

Advantages come with drawbacks:

✓ Instant global context through self-attention

✗ O(n^2) time-complexity w.r.t. number of tokens

✓ Modality agnostic representation of data

✗ Data-hungry compared to CNNs

Many frameworks for a solution…

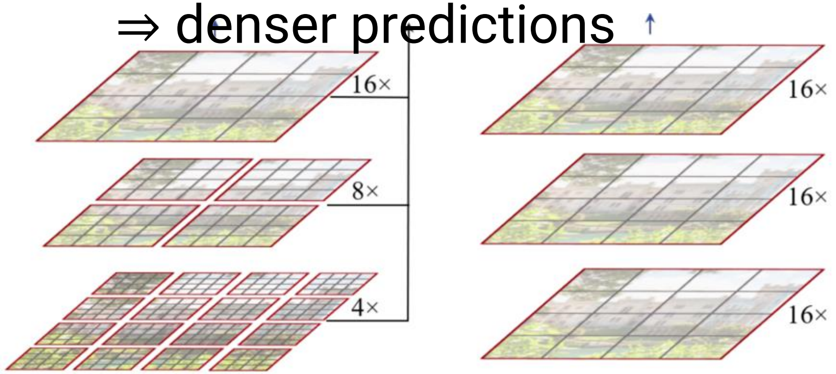
- Linear attention approx: *Linformer* (2020), *Performer* (2021), …
- Hardware-aware optim: *FlashAttention* (2022), …
- Compressed internal representation: *Perceiver* (2021), …

Transformers with Spatial Priors

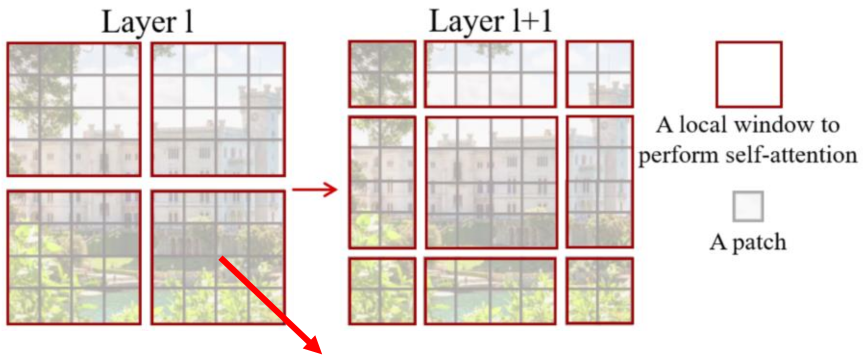
Shifted *win*dows (Swin) Transformer

- Self-attention within windows
  ⇒ O(n) w.r.t. nb of patches
- Smaller patches: 4x4 < ViT's 16x16
  ⇒ denser predictions

- Shifted windows
  ⇒ communication between windows



(a) Swin Transformer (ours) (b) ViT



Windows shift direction

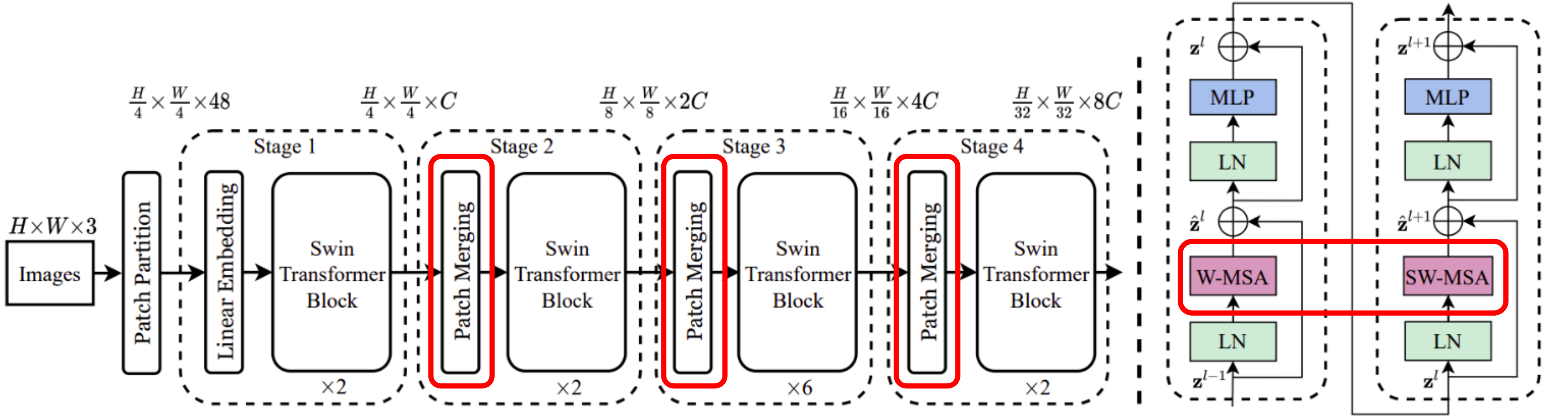
[Liu *et al.*, ICCV 2021]

Transformers with Spatial Priors

Shifted *win*dows (Swin) Transformer

- Patch merging = pooling in CNNs
- Transformer blocks: only change attention layers

Acronyms
W-MSA: Windows Multi-head Self-Attention
SW-MSA: Shifted W-MSA



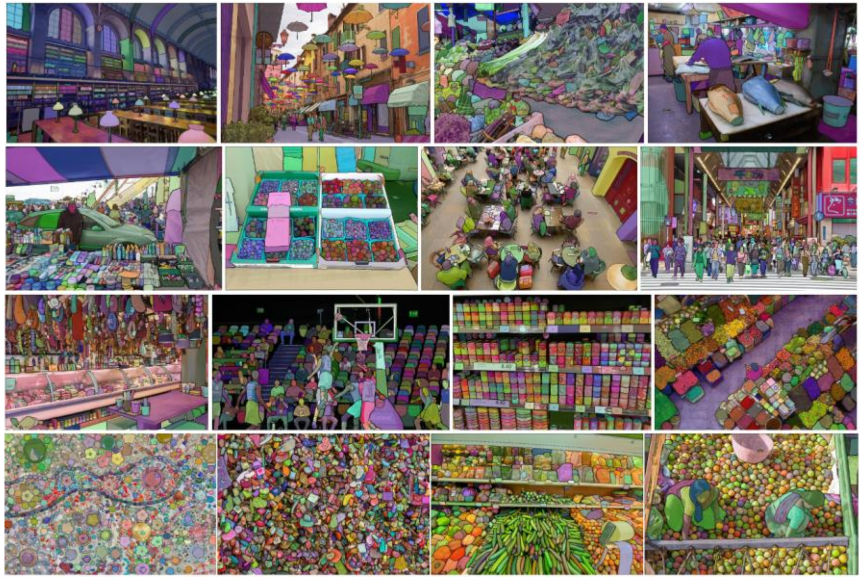
(a) Architecture (b) Two Successive Swin Transformer Blocks

[Liu *et al.*, ICCV 2021]

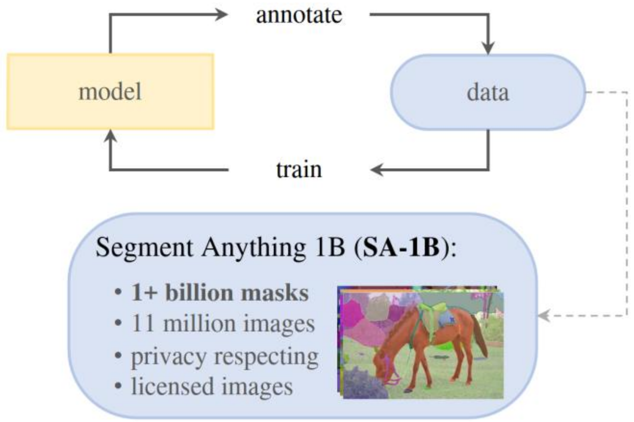
Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

- 2D natural images



Task	Classification	Segmentation
ViT	300 M	-
Swin Transformer	20 K	1.28 M
SAM	-	11 M images 1.1 B masks



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

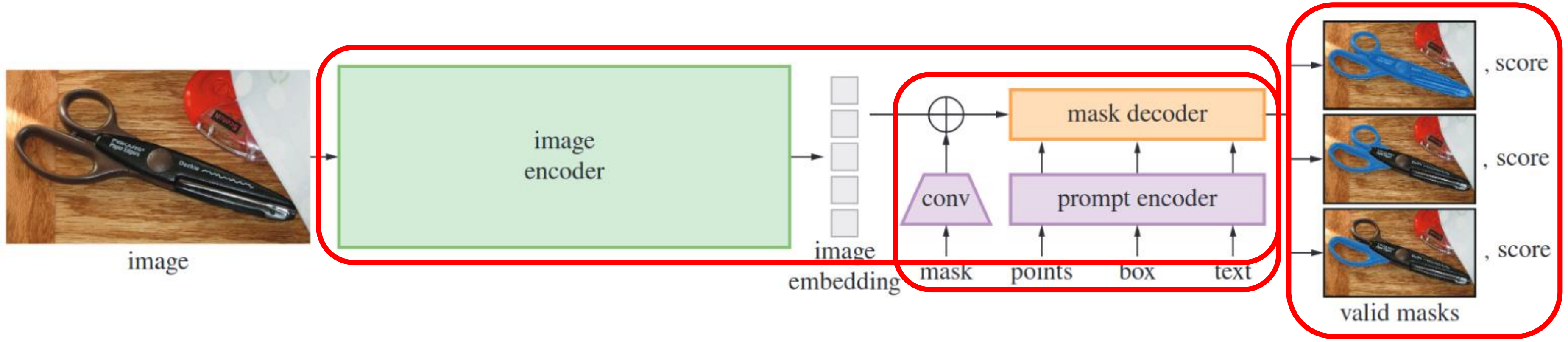
- **1+ billion masks**
- 11 million images
- privacy respecting
- licensed images

[Kirillov *et al.*, ICCV 2023]

Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

- Relatively simple architecture
- Interactive segmentation using prompts
- Accounts for ambiguous masks based on high-level prompt

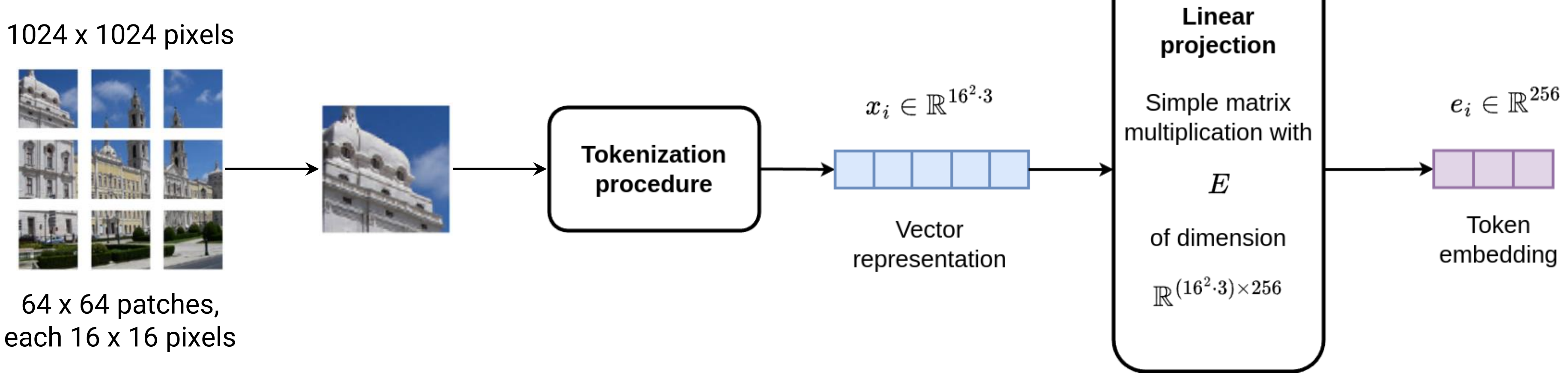


[Kirillov *et al.*, ICCV 2023]

Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

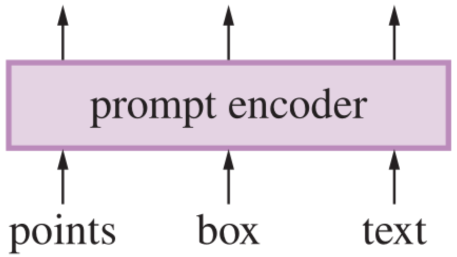
- Image encoder
  - Resize images to 1024 x 1024 pixels
  - Backbone: ViT-Huge with 16 x 16 pixels patches
  - Tokens: 256 dim.

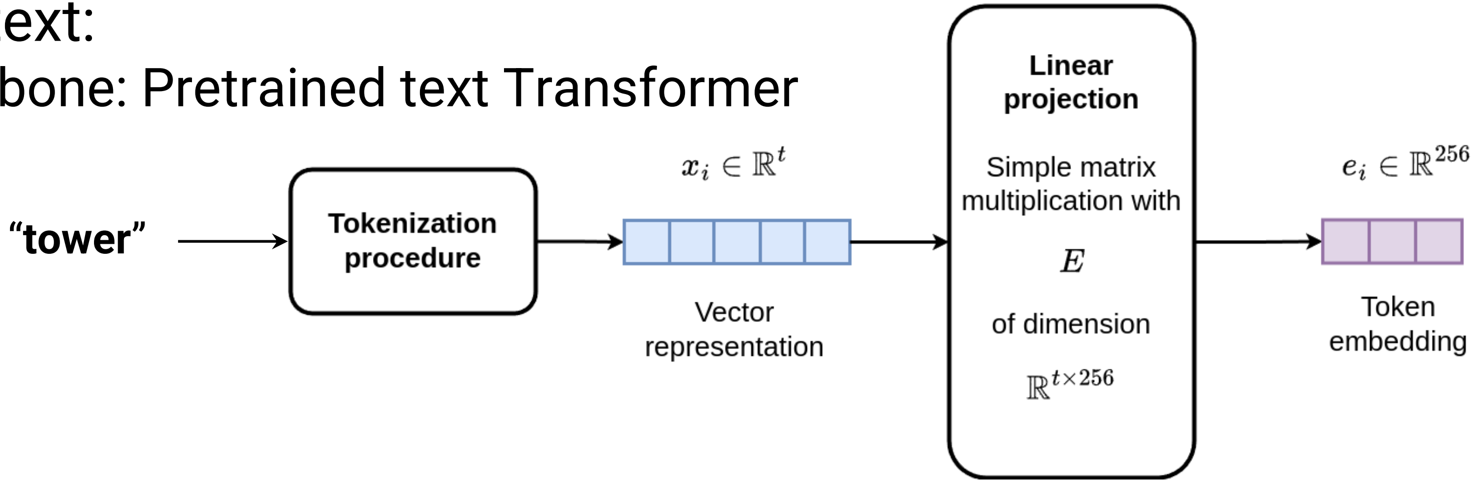

1024 x 1024 pixels
64 x 64 patches, each 16 x 16 pixels
Tokenization procedure
$x_i \in \mathbb{R}^{16^2 \cdot 3}$
Vector representation
Linear projection
Simple matrix multiplication with
E
of dimension
$\mathbb{R}^{(16^2 \cdot 3) \times 256}$
$e_i \in \mathbb{R}^{256}$
Token embedding

Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

- Prompt encoder
  - Different prompt types
  - Tokens: 256 dim.
- E.g. for text:
  - Backbone: Pretrained text Transformer

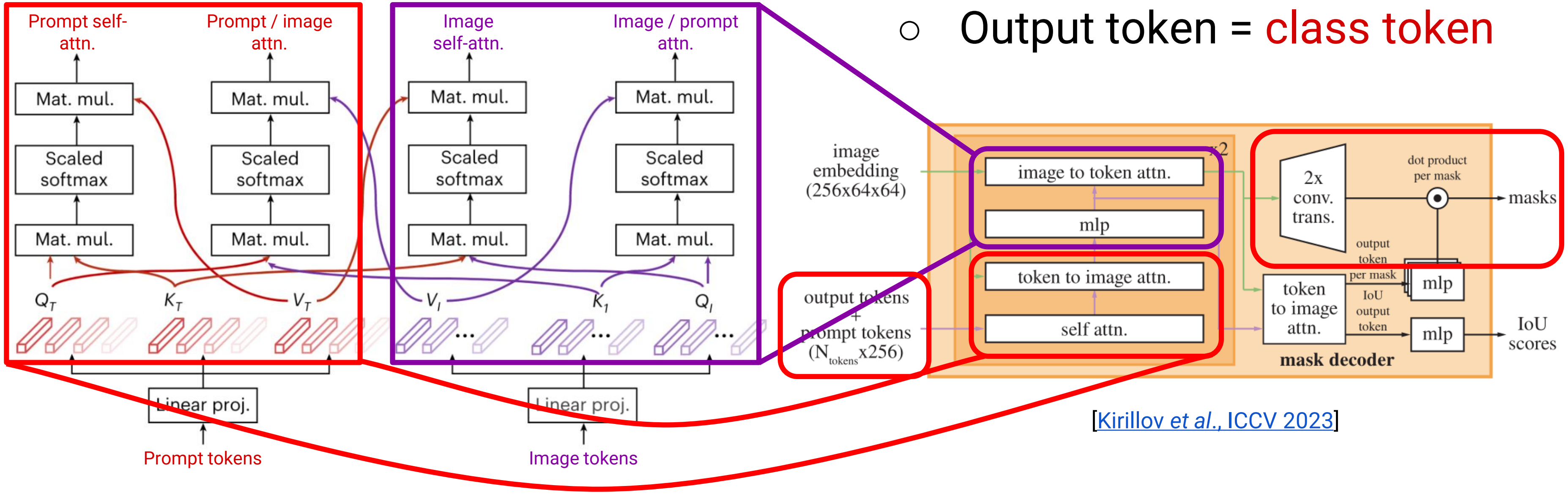




Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

- Mask decoder
  - Cross-attention: **Q, K, V** from two sources
  - Output token = class token

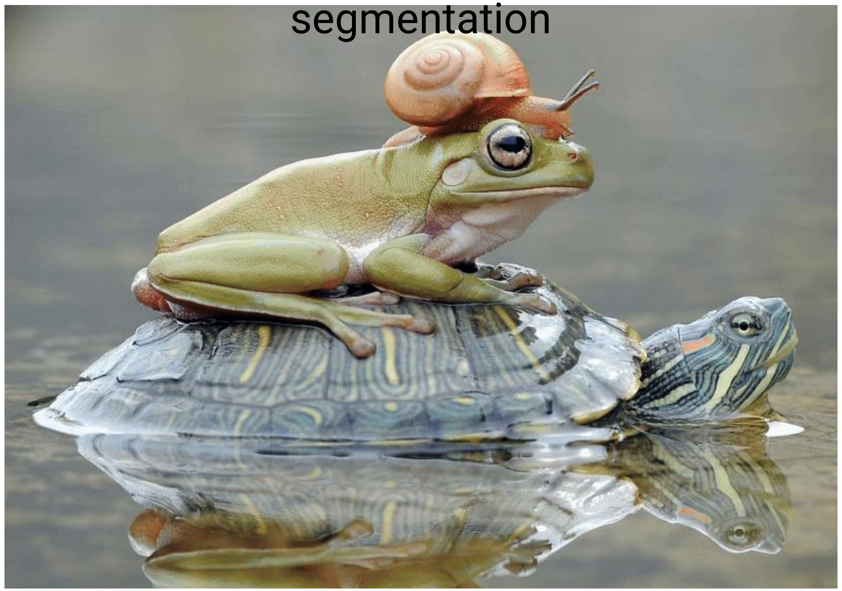


[Kirillov *et al.*, ICCV 2023]

Transformers at Scale: Foundation Models

Segment Anything Model (SAM)

User prompts ⇒ Interactive segmentation



Automatic prompts (e.g. regular grid) ⇒ Automatic segmentation

[Source: Segment Anything, Meta AI]

Alternative Paradigms

Is Attention All We Need?

✗ Transformer not implicitly suited for long context

Effectiveness / **efficiency** trade-off

L: Sequence length	Training	Inference
RNNs, LSTMs	**-Effective / +Efficient** Serial tokens computation: O(L)	Look at last step: O(L)
Transformers	**++Effective / -Efficient** Tokens x tokens: O(L^2) but **parallelizable**	Look at prev. tokens: O(L^2)
Mamba	**+Effective / ++Efficient** Serial tokens comput.: O(L) + **hardware optim.**	Look at last step: O(L)

What Are State Space Models?

[Gu *et al.*, ICLR 2022]

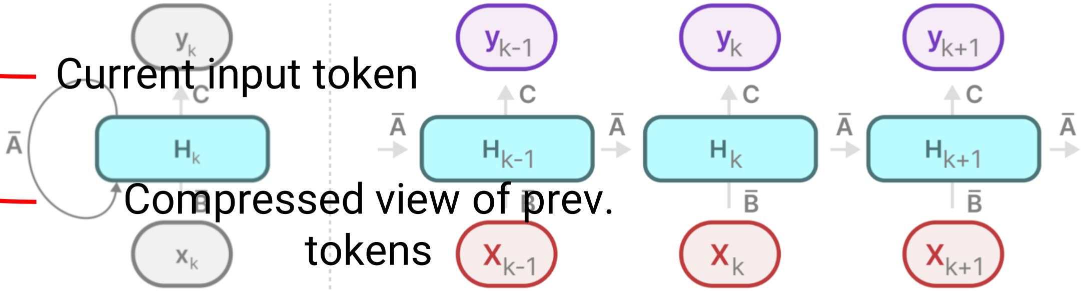
- General framework, including *Recurrent Neural Networks* (RNN)
  - **Internal state** ⇒ compressed view of previous tokens
  - **(Learnable) matrices** describe input / state / output interactions

$$h_{k+1} = \bar{\mathbf{A}} \boxed{h_k} + \bar{\mathbf{B}} \boxed{x_k}$$
$$y_k = \mathbf{C} \boxed{h_k}$$

Current input token

Compressed view of prev. tokens

State Space Models equations



[Source: A Visual Guide to Mamba and State Space Models]

- Commonly, $\mathbf{A}$ and $\mathbf{B} \perp\!\!\!\perp x_k$ (i.e. *linear time invariance*) ⇒
  - ✓ Parallelizable convolutions ⇒ fast!
  - ✗ Less expressive context ⇒ limits effectiveness

What Did Mamba Change in SSM?

[Gu and Dao, COLM 2024]

Improve theoretical **effectiveness**…

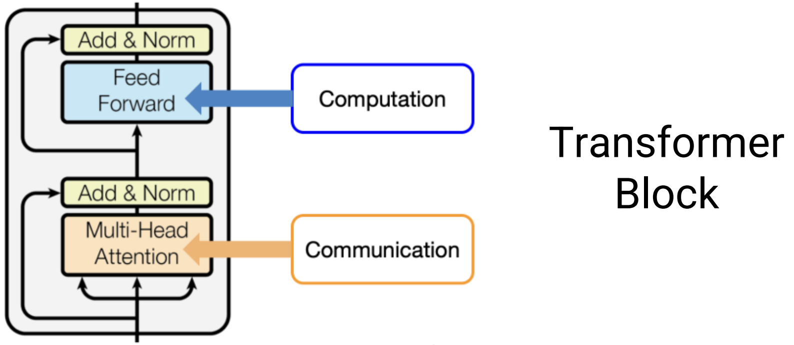
- **Selective** SSM
  - Make **B**, **C**, and **Δ** functions (i.e. linear proj.) of the input
    - Similar to **Q**, **K**, **V** projections
  - ✓ Able to store/forget specific inputs ⇒ **+effectiveness**
  - ✗ No convolutional representation

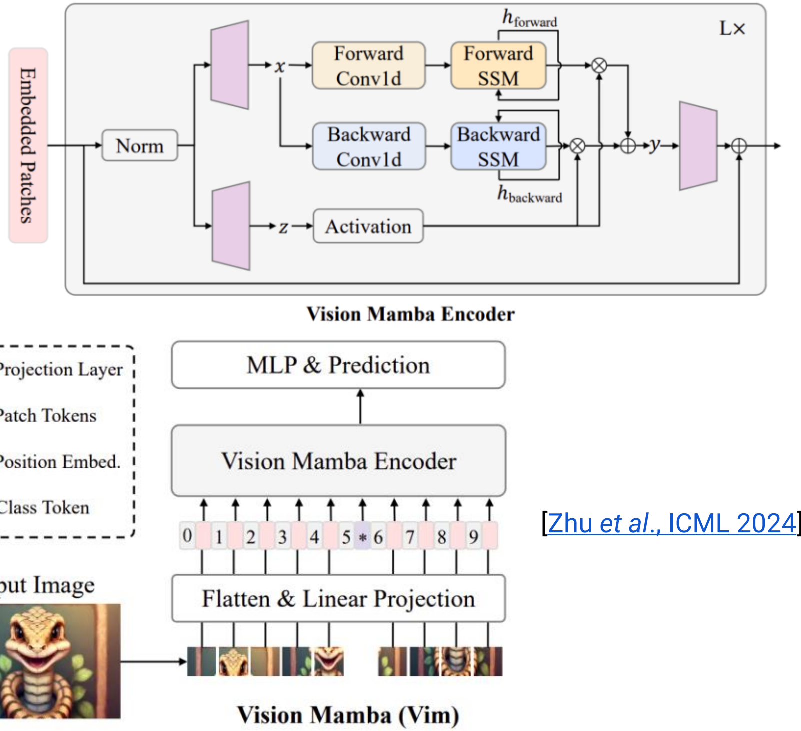
And practical implementation!

- **Parallel scan:** Compute matrix mul. in parts + combination
- **Kernel fusion:** Fuse steps to avoid unnecessary memory I/O
- **Recomputation:** Recompute interm. states rather than store them

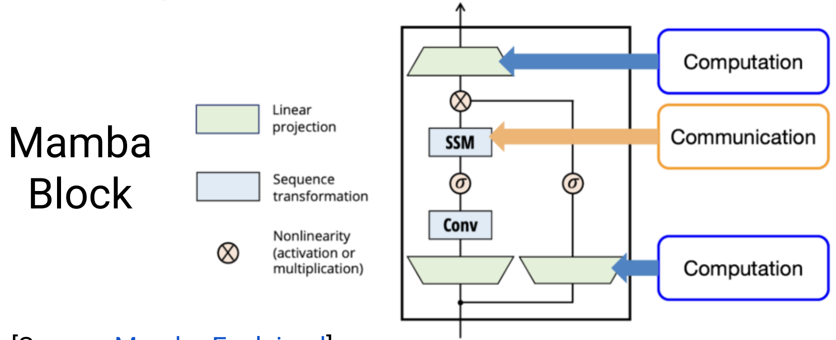
Encoder: Transformers vs Mamba

- Same 2-step process
- Encoder = Repeat blocks





[Zhu et al., ICML 2024]



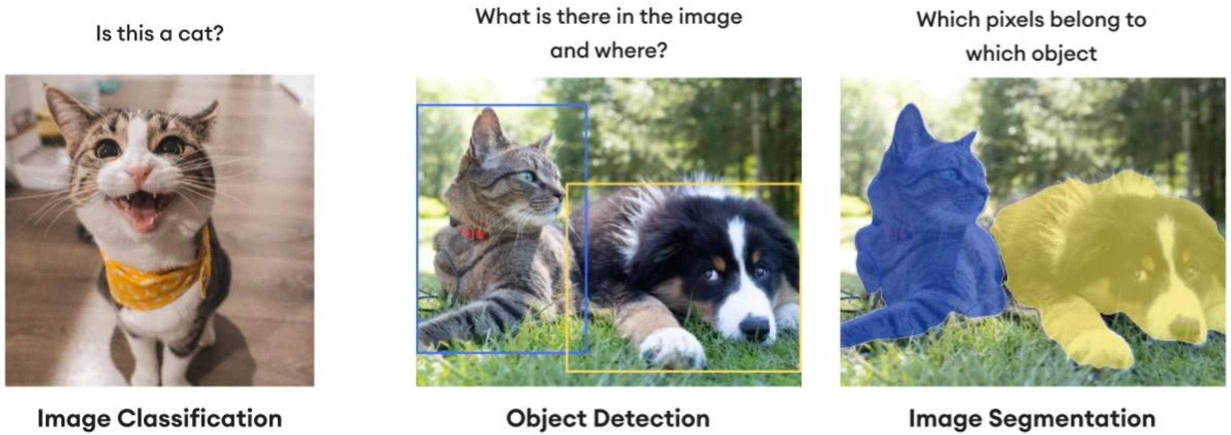
[Source: Mamba Explained]

Do We Really Need Mamba for Vision?

- *Transformers*: popularized framing tasks as **sequence modelling**
- *Mamba*: **address limitations** for autoregressive tasks / **long sequences** (> 2,000-4,000 tokens)
    - e.g. UltraLight VM-UNet: SOTA performance at 49K params

Is it useful for computer vision?

✗ Global tasks
✓ Dense tasks


Is this a cat?
What is there in the image and where?
Which pixels belong to which object
Image Classification
Object Detection
Image Segmentation

Takeaway Messages

- Transformers are the dominant (attention-based) paradigm
  - ✓ No input assumptions ⇒ easily adaptable to different data
  - ✓ Performance scaling with more compute and data

✗ Complexity ⇒ Time and space given sequence length
  - Motivated two main avenues of research
  - *Image specific*: architecture with **spatial priors**
  - *Generic tasks*: scale data for **foundation models**

- Alternative sequence models being explored…
  - ! Mind the task/data when choosing models

Thank you for your *attention* ;)

Questions?