# Deep Learning (M1)
## Attention mechanisms

Denis Coquenet

Université de Rennes

istic Informatique Électronique

# Goals of this course

## Knowledge

- Principle of attention mechanisms
- Transformer architecture
- Transformer attention formula

## Skills and know-how

- Integrate transformer attention in encoder / decoder
- Determine size and meaning of intermediate representations in attention mechanisms
- Analyze attention maps

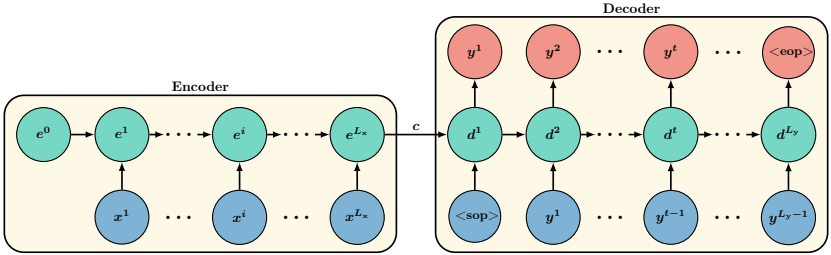
## Limitations of RNNs for large sequences

For feature extraction (encoder):

- Difficult to preserve information from start to end of the sequence

For seq2seq:

- The size of the context vector $c$ is fixed, no matter the sequence length (compression, information loss)



# Plan du cours

## Idea

From a static context vector:

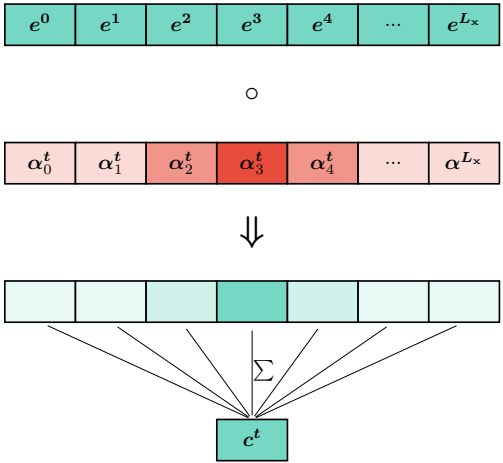
$$\boldsymbol{c} = f(\boldsymbol{x})$$

to a dynamic context vector:

$$\boldsymbol{c}^t = \sum_{i=1}^{L_x} \boldsymbol{\alpha}_i^t \boldsymbol{e}^i$$

with: $\sum_i \boldsymbol{\alpha}_i = 1$.

$\boldsymbol{\alpha}_i$ are called attention weights



## Encoder

Forward pass performed only once leading to $\{\boldsymbol{e}^1, ..., \boldsymbol{e}^{L_\times}\}$

## Decoder

Iterative process, at iteration $t$:

1. Compute score/energy for each encoded features $\boldsymbol{s}_i^t = a(...)$
2. Normalize these scores through softmax, leading to attention weights:

$$\boldsymbol{\alpha}_i^t = \frac{e^{\boldsymbol{s}_i^t}}{\sum_j e^{\boldsymbol{s}_j^t}}$$

3. Compute the context vector: $\boldsymbol{c}^t = \sum_i \boldsymbol{\alpha}_i^t \boldsymbol{e}^i$
4. Make prediction for iteration $t$: $\hat{\boldsymbol{y}}^t = b(\boldsymbol{c}^t)$

### Content-based attention

The scores are based on features and previous context: $\boldsymbol{s}_i^t = a(\boldsymbol{c}^{t-1}, \boldsymbol{e}^i)$

### Location-based attention

The scores are based on previous attention weights and previous context: $\boldsymbol{s}_i^t = a(\boldsymbol{c}^{t-1}, \boldsymbol{\alpha}^{t-1})$

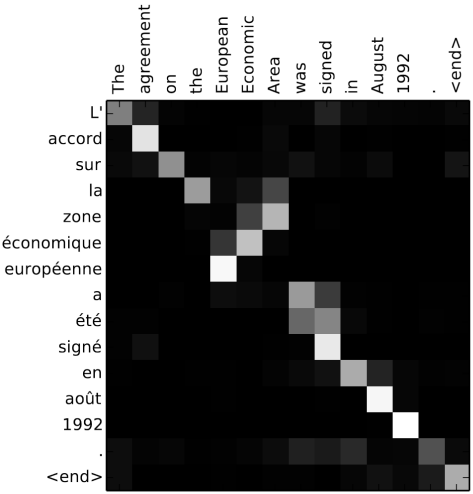
### Hybrid attention

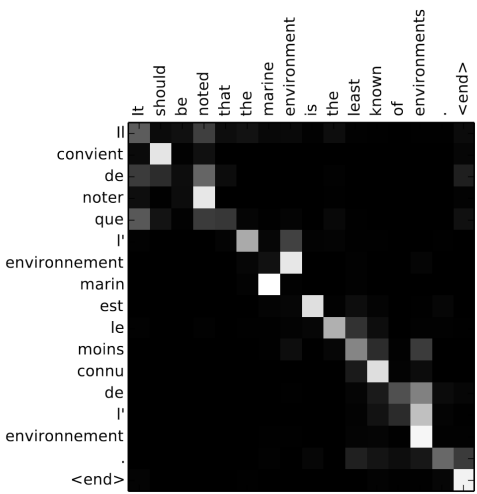
Generally, both approaches are mixed together: $\boldsymbol{s}_i^t = a(\boldsymbol{c}^{t-1}, \boldsymbol{\alpha}^{t-1}, \boldsymbol{e}^i)$

### Example

$\boldsymbol{s}_i^t = \boldsymbol{v}^T \tanh(\boldsymbol{W}_c \boldsymbol{c}^{t-1} + \boldsymbol{W}_\alpha \boldsymbol{\alpha}^{t-1} + \boldsymbol{W}_e \boldsymbol{e}^i)$

# Attention maps for encoder-decoder attention




Images from [1]

➤ Size: target sequence length × source sequence length

## Goal

Embed a token as a weighted sum of the other tokens of the sequence
Input: a set of token embeddings $\{\boldsymbol{e}^1, ..., \boldsymbol{e}^{L_x}\}$
Output: a new set of token embeddings $\{\boldsymbol{o}^1, ..., \boldsymbol{o}^{L_x}\}$

## For each token $\boldsymbol{e}^i$

1. Compute a score with respect to each token: $\boldsymbol{s}^i_j = a(\boldsymbol{e}^i, \boldsymbol{e}^j)$
2. Normalize the scores:
$$\boldsymbol{\alpha}^i_j = \frac{e^{\boldsymbol{s}^i_j}}{\sum_k e^{\boldsymbol{s}^i_k}}$$
3. Compute the context vector: $\boldsymbol{c}^i = \sum_j \boldsymbol{\alpha}^i_j \boldsymbol{e}^j$
4. Compute the new token representation: $\boldsymbol{o}^i = b(\boldsymbol{c}^i)$

➤ All $\boldsymbol{o}^i$ can be computed in parallel

Let's assume an input sequence $\boldsymbol{e} = [\boldsymbol{e}^1, ..., \boldsymbol{e}^{L_x}]$ with $\boldsymbol{e}^i \in \mathbb{R}^C$ and $\boldsymbol{W} \in \mathbb{R}^{C_o \times C}$.

- What is the size of $\boldsymbol{s}^i, \boldsymbol{\alpha}^i, \boldsymbol{c}^i$ and $\boldsymbol{o}^i$?

$$\boldsymbol{s}^i_j = \sum_k \boldsymbol{e}^i_k \boldsymbol{e}^j_k$$

$$\boldsymbol{\alpha}^i_j = \frac{e^{\boldsymbol{s}^i_j}}{\sum_k e^{\boldsymbol{s}^i_k}}$$

$$\boldsymbol{c}^i = \sum_j \boldsymbol{\alpha}^i_j \boldsymbol{e}^j$$

$$\boldsymbol{o}^i = \boldsymbol{W}\boldsymbol{c}^i$$

Let's assume an input sequence $\boldsymbol{e} = [\boldsymbol{e}^1, ..., \boldsymbol{e}^{L_x}]$ with $\boldsymbol{e}^i \in \mathbb{R}^C$

- There are $L_x$ tokens
- For a given token $i$, one score per token $j$: $\boldsymbol{s}^i \in \mathbb{R}^{L_x}$
- For a given token, we want one attention weight for each token: $\boldsymbol{\alpha}^i \in \mathbb{R}^{L_x}$
- Context vector is a weighted sum of vectors of size $C$: $\boldsymbol{c}^i \in \mathbb{R}^C$
- Fully connected layer applied to the context vector: $\boldsymbol{o}^i \in \mathbb{R}^{C_o}$

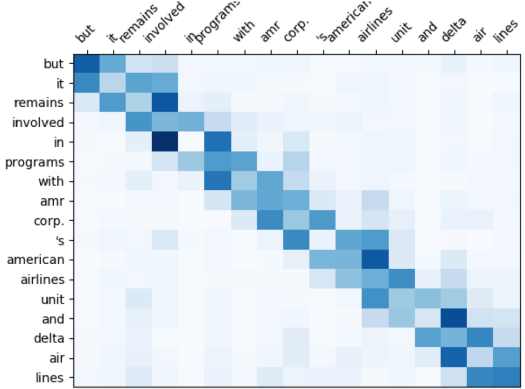
$$\boldsymbol{s}^i_j = \sum_k \boldsymbol{e}^i_k \boldsymbol{e}^j_k$$

$$\boldsymbol{\alpha}^i_j = \frac{e^{\boldsymbol{s}^i_j}}{\sum_k e^{\boldsymbol{s}^i_k}}$$

$$\boldsymbol{c}^i = \sum_j \boldsymbol{\alpha}^i_j \boldsymbol{e}^j$$

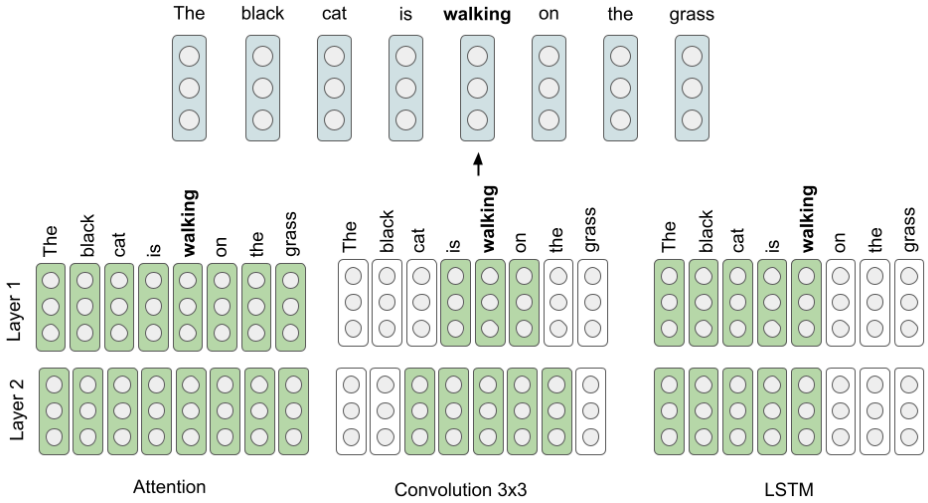
$$\boldsymbol{o}^i = \boldsymbol{W}\boldsymbol{c}^i$$

# Attention map for self-attention

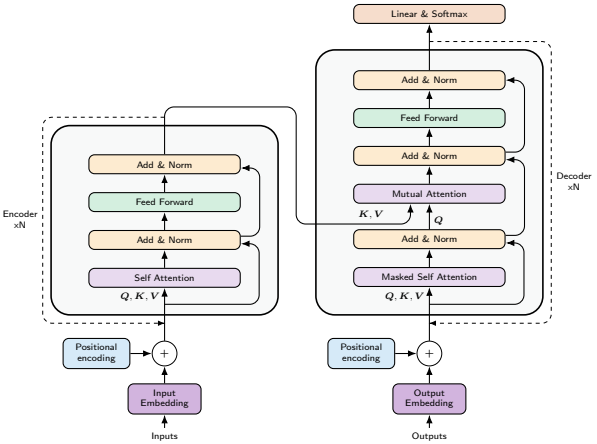



➤ Size: source sequence length $\times$ source sequence length

# Context modeling comparison



# Plan du cours




Two main blocks:

- Transformer encoder
- Transformer decoder

**One-shot encoding**

$$\boldsymbol{f} = \text{encoder}(\boldsymbol{x} + \text{PE})$$

**Iterative decoding**

$$\boldsymbol{c}^t = \text{decoder}(\boldsymbol{f}, \{\hat{\boldsymbol{y}}^0, ..., \hat{\boldsymbol{y}}^{t-1}\} + \text{PE})$$

$$\hat{\boldsymbol{y}}^t = \arg\max(\boldsymbol{W}\boldsymbol{c}^t)$$

➤ Mainly relies on the Query-Key-Value attention paradigm

## Concept

Goal: retrieve the useful information among a collection of data
Inputs: a *query* and a set of data

## Approach

- Compute a similarity score between the query and all the data
- Generate an answer based on the most relevant data

➤ Two representations of the data are used: *keys* for comparison with the query, and *values* for the answer

## Analogy with library search engine

Query: keywords; Keys: book titles; Values: book contents

## Issue

Gradient propagation for hard attention (select only the most relevant item)

➤ Soft attention (weighted sum of items)

## Scaled Dot-Product Attention

$\boldsymbol{q} \in \mathbb{R}^d$: a query vector

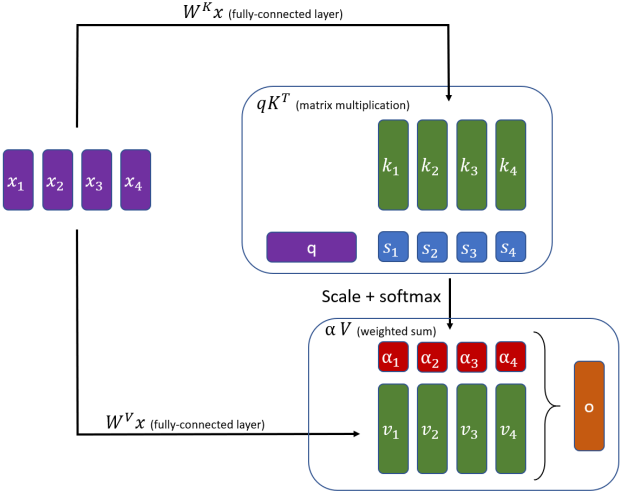
$\boldsymbol{K} \in \mathbb{R}^{L \times d}$: a matrix gathering key vectors for each available item

$\boldsymbol{V} \in \mathbb{R}^{L \times d}$: a matrix gathering values vectors for each available item

$$\text{Attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \underbrace{\text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{K}^T}{\sqrt{d}}\right)}_{\alpha} \boldsymbol{V} = \boldsymbol{c} \quad (\in \mathbb{R}^d)$$

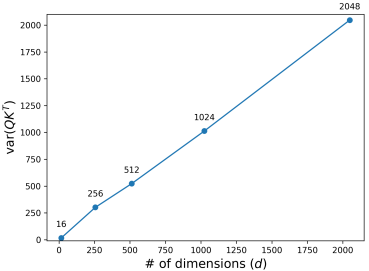
➤ Dot product as distance function

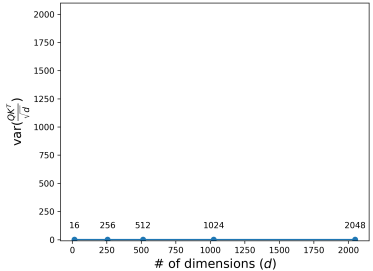
# Illustration of the computations

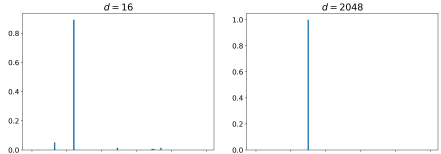


# Why scaling by $\frac{1}{\sqrt{d}}$?

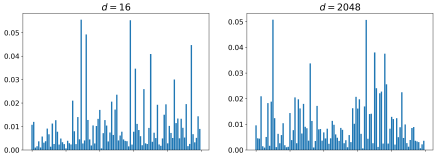
For 1 query and 10,000 keys (from normal distribution):




Softmax over the first 100 keys




Softmax over the first 100 keys

➤ Queries, Keys and Values are from the same source

**Query-key-value for feature extraction**

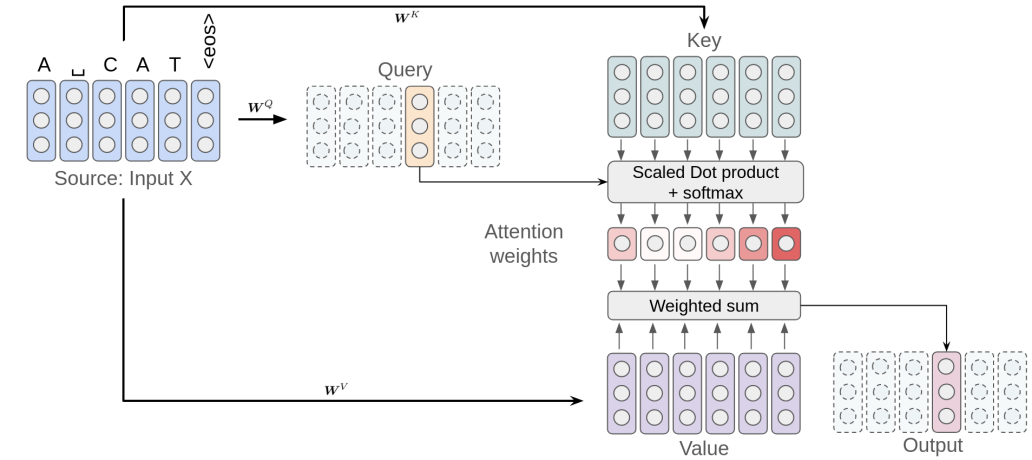
$\boldsymbol{X}$: the latent representation of an input sequence.

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q} (\in \mathbb{R}^{L_x \times d})$$
$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K} (\in \mathbb{R}^{L_x \times d})$$
$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V} (\in \mathbb{R}^{L_x \times d})$$

$$\boldsymbol{Y} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{d}}\right)\boldsymbol{V}$$

$\boldsymbol{W}^{Q}, \boldsymbol{W}^{K}$ and $\boldsymbol{W}^{V}$ are matrices of trainable weights (fully-connected layers)

# Self-attention




- Output can be computed in parallel through matrix multiplications
- Output length = Query length ( = Key length = Value length)

## $h$ self-attention in parallel

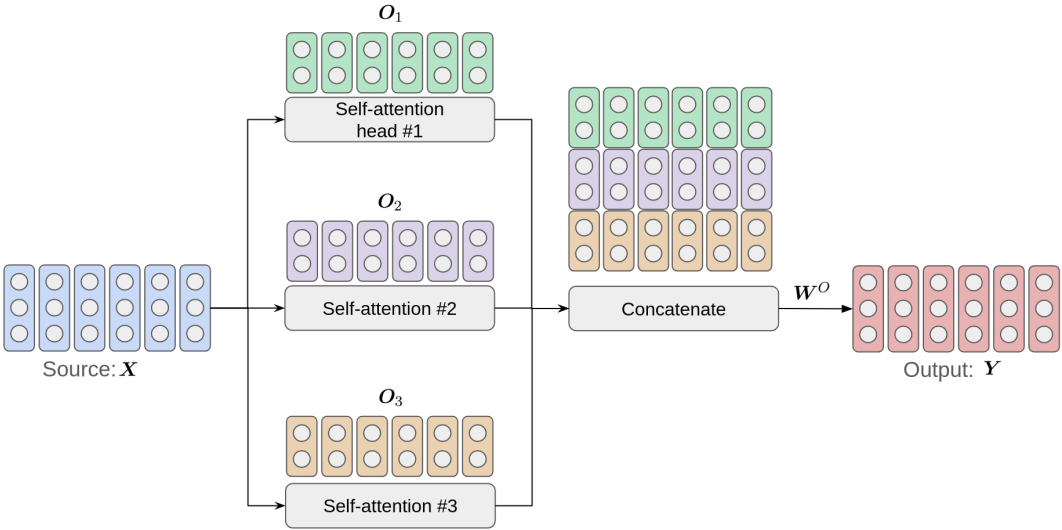
$$\begin{aligned} \boldsymbol{Q}_1 &= \boldsymbol{X}\boldsymbol{W}_1^Q \quad \dots \quad \boldsymbol{Q}_h = \boldsymbol{X}\boldsymbol{W}_h^Q \\ \boldsymbol{K}_1 &= \boldsymbol{X}\boldsymbol{W}_1^K \quad \dots \quad \boldsymbol{K}_h = \boldsymbol{X}\boldsymbol{W}_h^K \\ \boldsymbol{V}_1 &= \boldsymbol{X}\boldsymbol{W}_1^V \quad \dots \quad \boldsymbol{V}_h = \boldsymbol{X}\boldsymbol{W}_h^V \end{aligned}$$

$$\boldsymbol{O}_1 = \text{softmax}\left(\frac{\boldsymbol{Q}_1\boldsymbol{K}_1^T}{\sqrt{d}}\right)\boldsymbol{V}_1 \quad \dots \quad \boldsymbol{O}_h = \text{softmax}\left(\frac{\boldsymbol{Q}_h\boldsymbol{K}_h^T}{\sqrt{d}}\right)\boldsymbol{V}_h$$

$$\boldsymbol{Y} = \text{concat}(\boldsymbol{O}_1, \dots, \boldsymbol{O}_h)\boldsymbol{W}^O$$

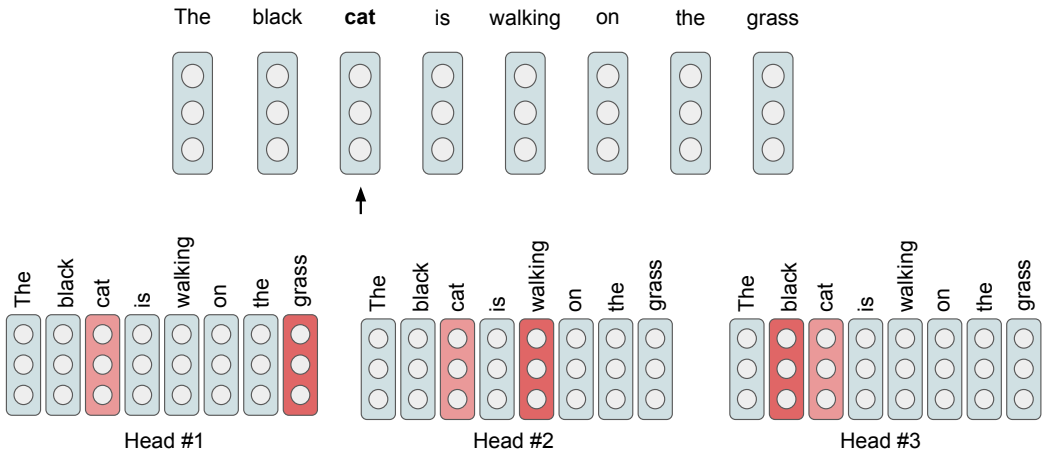
Heads can be seen as the equivalent of convolutional kernels

# Multi-head Self-Attention (MSA)



## Intuition

Each head is specialized for a specific query (*e.g.*, action, location, description)



## Issue with Multi-head Self Attention (MSA)

Self-attention is equivariant to input permutation:
$\text{MSA}([\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3]) = [\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3] \Rightarrow \text{MSA}([\boldsymbol{x}_2, \boldsymbol{x}_1, \boldsymbol{x}_3]) = [\boldsymbol{y}_2, \boldsymbol{y}_1, \boldsymbol{y}_3]$

The position information is lost whereas it is crucial

➤ A word/sentence is a sequence of characters, not a set of characters

## Goal

Inject positional information to preserve the sequentiality

## Solution

Additive positional encoding:

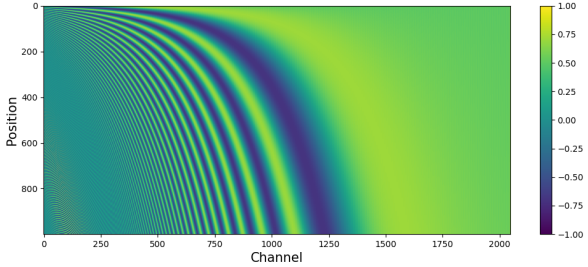
➤ Input of transformer encoder: $\boldsymbol{I} = \boldsymbol{X} + \text{PE}$

## A combination of sines and cosines

PE$(p, k)$: $k^{\text{th}}$ value of positional embedding vector for position $p$.

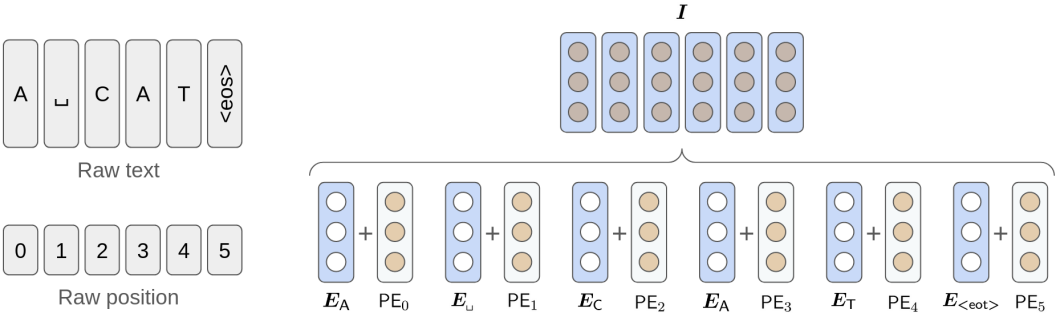
$$\text{PE}(p, 2k) = \sin(w_k \cdot p)$$
$$\text{PE}(p, 2k+1) = \cos(w_k \cdot p)$$

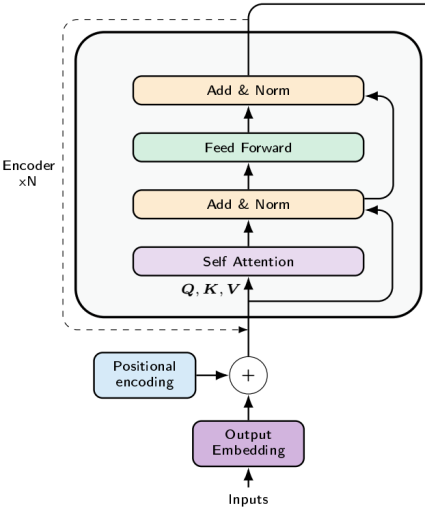
$\forall k \in [0, d/2]$, with $w_k = 1/10000^{2k/d}$



## Advantages

- Does not depend on the input length
- Encoding unique for each position
- Periodicity of sine/cosine could help to adapt to unseen position
- Bonus: no additional trainable parameters






## A stack of 6 transformer encoder layers

- Multi-head Self Attention
  - ➤ Global context modeling
- Add: residual connections
  - ➤ Multi-scale representation
  - ➤ Reinforce identity
- Norm: layer normalization
  - ➤ Stabilize training (range of values)
- Feed Forward: 2 fully-connected layers
  - ➤ Local projection

Input sequence of $L_x$ token embeddings: $\boldsymbol{X} \in \mathbb{R}^{L \times d}$

A simplified version of the attention layer, keeping the same dimension:
$\boldsymbol{Y} = (\boldsymbol{X}\boldsymbol{W}^{\mathsf{Q}})(\boldsymbol{X}\boldsymbol{W}^{\mathsf{K}})^T(\boldsymbol{X}\boldsymbol{W}^{\mathsf{V}})\boldsymbol{W}^{\mathsf{O}}$

$\boldsymbol{W}^{\mathsf{Q}} \in \mathbb{R}^{d \times d}, \boldsymbol{W}^{\mathsf{K}} \in \mathbb{R}^{d \times d}, \boldsymbol{W}^{\mathsf{V}} \in \mathbb{R}^{d \times d}, \boldsymbol{W}^{\mathsf{O}} \in \mathbb{R}^{d \times d} \rightarrow 4d^2$ parameters

$\boldsymbol{X}\boldsymbol{W}^{\mathsf{i}}$: from $L \times d$ to $L \times d$; $2d^2L$ FLOPs ($\times 3$)

$\boldsymbol{S} = (\boldsymbol{X}\boldsymbol{W}^{\mathsf{Q}})(\boldsymbol{X}\boldsymbol{W}^{\mathsf{K}})^T$: output $L \times L$; $2dL^2$ FLOPs

$\boldsymbol{O} = \boldsymbol{S}(\boldsymbol{X}\boldsymbol{W}^{\mathsf{V}})$: output $L \times d$; $2dL^2$ FLOPs

$\boldsymbol{Y} = \boldsymbol{O}\boldsymbol{W}^{\mathsf{O}}$: output $L \times d$; $2d^2L$ FLOPs

- $8d^2L + 4dL^2$ FLOPs
- Quadratic complexity with respect to the input length

➤ Queries from target domain, and keys and values from source domain

## General case

Inputs: $\boldsymbol{X}$: the features from the encoded input sequence (source domain),
$\tilde{\boldsymbol{z}}$: the query sequence (*e.g.*, a question)
$\boldsymbol{z}$: the query vector ($\boldsymbol{z} = g(\tilde{\boldsymbol{z}})$).

$$\boldsymbol{q} = \boldsymbol{z}\boldsymbol{W}^Q (\in \mathbb{R}^d)$$
$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K (\in \mathbb{R}^{L_x \times d})$$
$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V (\in \mathbb{R}^{L_x \times d})$$

$$\boldsymbol{c} = \text{Attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \underbrace{\text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{K}^T}{\sqrt{d}}\right)}_{\alpha}\boldsymbol{V}$$

Prediction: $\hat{\boldsymbol{y}} = h(\boldsymbol{c})$, with $g$ and $h$ parametric functions

➤ Queries from target domain, and keys and values from source domain

**The iterative decoding process case**

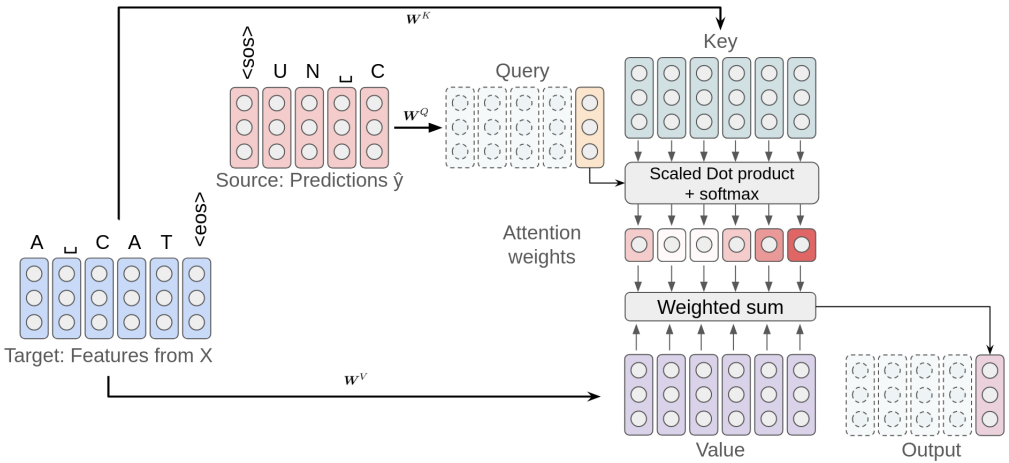
Inputs: $\boldsymbol{X}$: the features from the encoded input sequence (source domain),
$\tilde{\boldsymbol{z}} = \hat{\boldsymbol{y}}^{0:t-1}$: an ouput sequence (target domain)
$\boldsymbol{z}^t$: a query vector at iteration $t$ ($\boldsymbol{z}^t = g(\hat{\boldsymbol{y}}^0, ..., \hat{\boldsymbol{y}}^{t-1})$).

$$\boldsymbol{q}^t = \boldsymbol{z}^t \boldsymbol{W}^Q (\in \mathbb{R}^d)$$
$$\boldsymbol{K} = \boldsymbol{X} \boldsymbol{W}^K (\in \mathbb{R}^{L_x \times d})$$
$$\boldsymbol{V} = \boldsymbol{X} \boldsymbol{W}^V (\in \mathbb{R}^{L_x \times d})$$

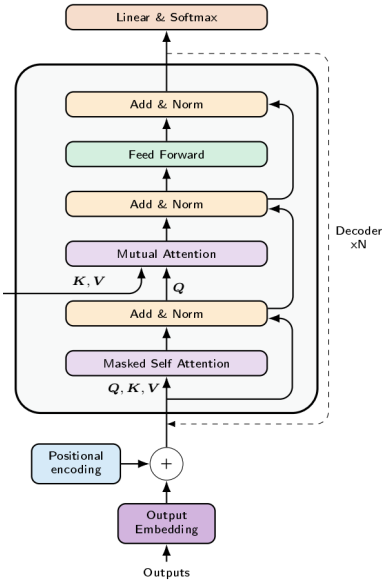
$$\boldsymbol{c}^t = \text{Attention}(\boldsymbol{q}^t, \boldsymbol{K}, \boldsymbol{V}) = \underbrace{\text{softmax}\left(\frac{\boldsymbol{q}^t \boldsymbol{K}^T}{\sqrt{d}}\right)}_{\alpha^t} \boldsymbol{V}$$

New prediction: $\hat{\boldsymbol{y}}^t = h(\boldsymbol{c}^t)$, with $g$ and $h$ parametric functions

# Cross attention attention



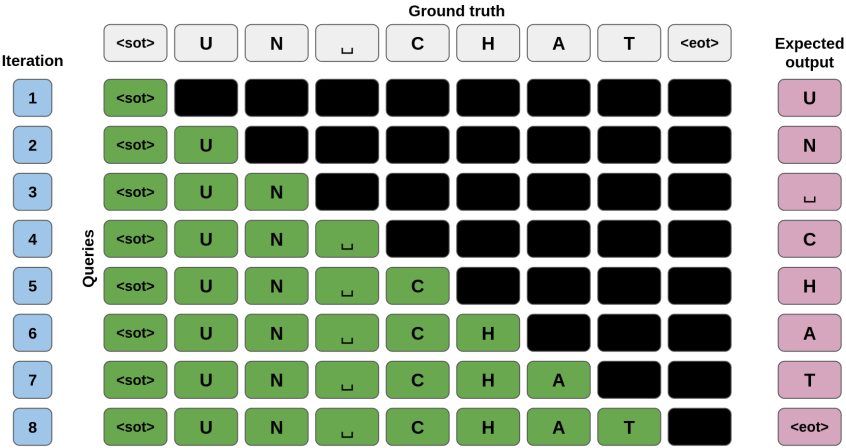

➤ Output length = Query length ($\neq$ Key length = Value length)




## A stack of 6 transformer decoder layers

- Multi-head Masked Self Attention
  ➤ Global query modeling
- Add: residual connections
  ➤ Multi-scale representation
  ➤ Reinforce identity
- Norm: layer normalization
  ➤ Stabilize training (range of values)
- Multi-head mutual attention
  ➤ Look for relevant information from source
- Feed Forward: 2 fully-connected layers
  ➤ Local projection

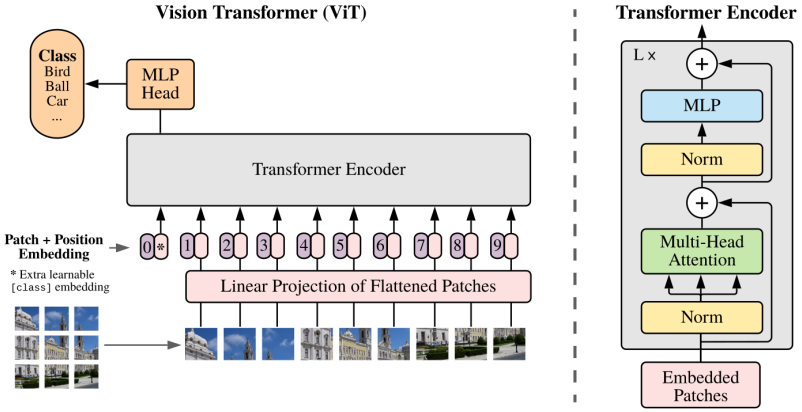
➤ Parallelize decoding process at training time using ground truth



## Conclusion

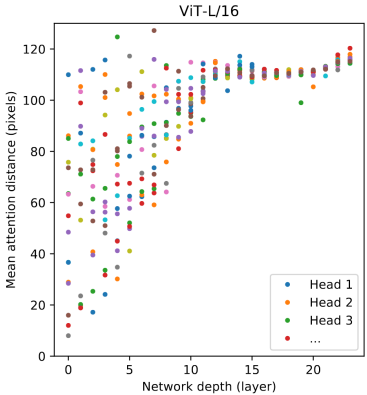
- A modular approach: expressivity can be enhanced by adding more encoder/decoder layers
  ➤ But slower training/inference and more required data
- Self attention: a way to model global context without sequential computations (at each layer, each token can attend to all the other tokens)
  ➤ But a quadratic complexity
- Parameter-free additive positional encoding
  ➤ But the relation between positions must be learned through training
- Defined for Natural Language Processing: SOTA architecture
  ➤ How to adapt it for computer vision?

# Transformer for image classification: Vision Transformer (2021) [3]




Input image: $224 \times 224$, patch size: $16 \times 16 \Rightarrow 196$ patches
Introduction of a trainable class token used for prediction


Attention map for the class token



| Architecture | Top-1 accuracy (%) on ImageNet | # params (M) |
|---|---|---|
| ResNet-50 | 79.26 | 26 |
| ResNet-101 | 80.13 | 45 |
| ResNet-152 | **80.62** | 60 |
| ViT-B/16 | 77.91 | 86 |
| ViT-B/32 | 73.38 | 86 |
| ViT-L/16 | 76.53 | 307 |
| ViT-L/32 | 71.16 | 307 |

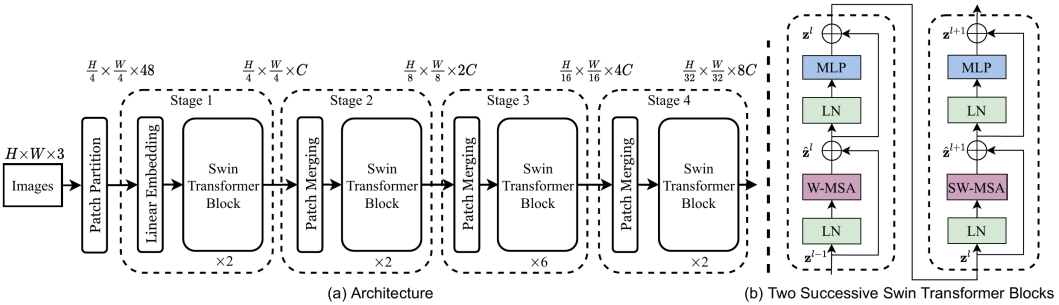
Attention-based architectures require more data to exploit their full potential:

➤ Spatial relations between patches must be learned from scratch

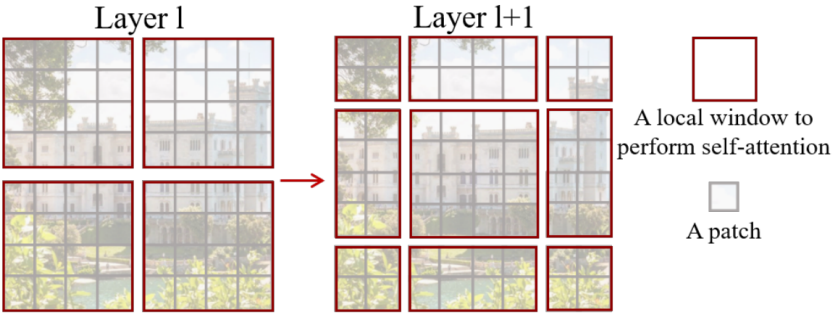
Performance on ImageNet (fine-tuning included)

| Architecture | Top-1 accuracy (%) on ImageNet | |
|---|---|---|
| | Pre-trained on ImageNet | Pre-trained on JFT-300M |
| ResNet-50 | 79.26 | ✗ |
| ResNet-101 | 80.13 | ✗ |
| ResNet-152 | 80.62 | ✗ |
| ViT-B/16 | 77.91 | 84.15 |
| ViT-B/32 | 73.38 | 80.73 |
| ViT-L/16 | 76.53 | 87.12 |
| ViT-L/32 | 71.16 | 84.37 |
| ViT-H/14 | ✗ | **88.04** |

➤ Hierarchical Vision Transformer using Shifted windows (Swin)




Patch size: $4 \times 4$
Patch merging: $2 \times 2$ adjacent patches are merged together
(S)W-MSA: (Shifted) Window Multi Self-Attention
➤ Idea: perform attention on patch windows (locally), in parallel




**Modeling global context with shifted windows**

Windows are shifted from one layer to another to propagate the information from one window to its neighbours

**Reminder: simplified self-attention**

Number of FLOPs: $8d^2L + 4dL^2$
With $d$ the number of dimensions and $L$ the sequence length

Compute the number of FLOPs for a latent representation of an image of size $(H = 1024, W = 1024, C = 256)$

- for a traditional self-attention layer
- for the shifted windows approach (window size: $8 \times 8$, first step only)

➤ Remark: the number of dimensions is preserved (from 256 to 256)

## Reminder: simplified self-attention

Number of FLOPs: $8d^2L + 4dL^2$
With $d$ the number of dimensions and $L$ the sequence length

## For a traditional self-attention layer

Flatten: $L = HW = 1024 \times 1024 = 1,048,576$
➤ $8 \times 256^2 \times 1,048,576 + 4 \times 256 \times 1,048,576^2 = 1.1 \times 10^{15}$ FLOPs

## For a shifted-window self-attention layer

Window size: $8 \times 8$
Flatten: $L = HW = 8 \times 8 = 64$
➤ $8 \times 256^2 \times 64 + 4 \times 256 \times 64^2 = 37.7 \times 10^6$ FLOPs per window

Number of windows: $\frac{1024}{8} \times \frac{1024}{8} = 16,384$
➤ $16,384 \times 37.7 \times 10^6 = 0.6 \times 10^9$ FLOPs

| method | image size | #param. | FLOPs | throughput (image / s) | ImageNet top-1 acc. |
|---|---|---|---|---|---|
| R-101x3 [38] | $384^2$ | 388M | 204.6G | - | 84.4 |
| R-152x4 [38] | $480^2$ | 937M | 840.5G | - | 85.4 |
| ViT-B/16 [20] | $384^2$ | 86M | 55.4G | 85.9 | 84.0 |
| ViT-L/16 [20] | $384^2$ | 307M | 190.7G | 27.3 | 85.2 |
| Swin-B | $224^2$ | 88M | 15.4G | 278.1 | 85.2 |
| Swin-B | $384^2$ | 88M | 47.0G | 84.7 | 86.4 |
| Swin-L | $384^2$ | 197M | 103.9G | 42.1 | 87.3 |

Pre-training on ImageNet-22k

# Plan du cours

## Create artificial examples

- Data augmentation
- Synthetic data

## Benefit from other annotated datasets

- Transfer learning
- Fine-tuning
- Multi-tasking

## Benefit from unlabeled data

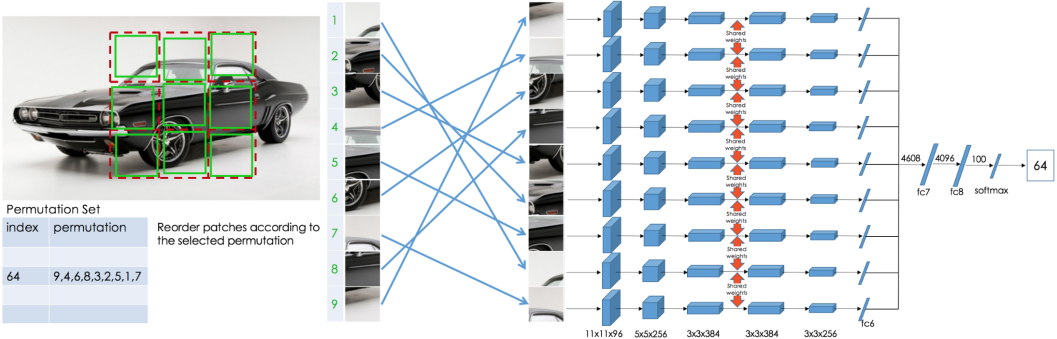
- Semi-supervised learning
- **Self-supervised learning**

**The goal is to solve a pretext task**

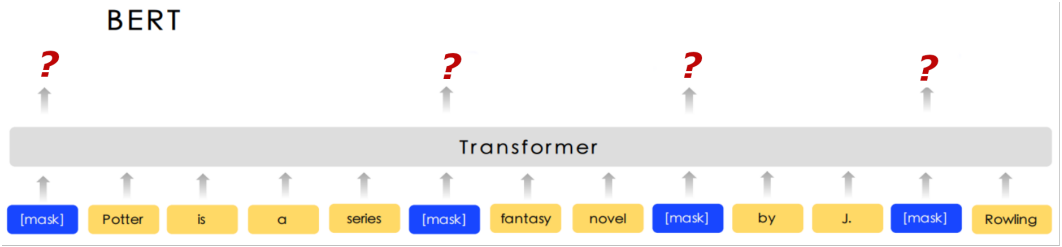
Some examples:

- Solve Jigsaw puzzles
- Reconstruct the input image:
  (Variational) Auto-encoders, Masked auto-encoders
- Contrastive learning: SimCLR, CLIP
  Learning from positive and negative samples

➤ Find the right permutation: classification formulation

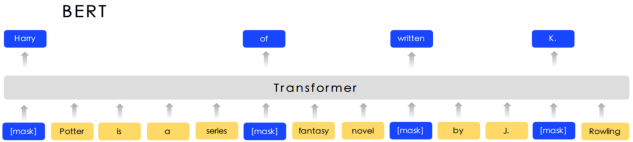


➤ Two self-supervised strategies

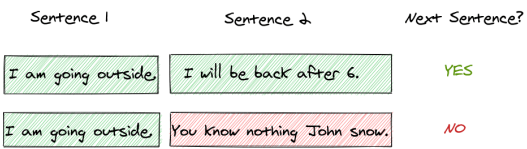


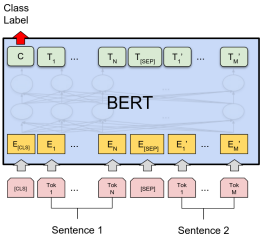

Masked LM

➤ Two self-supervised strategies



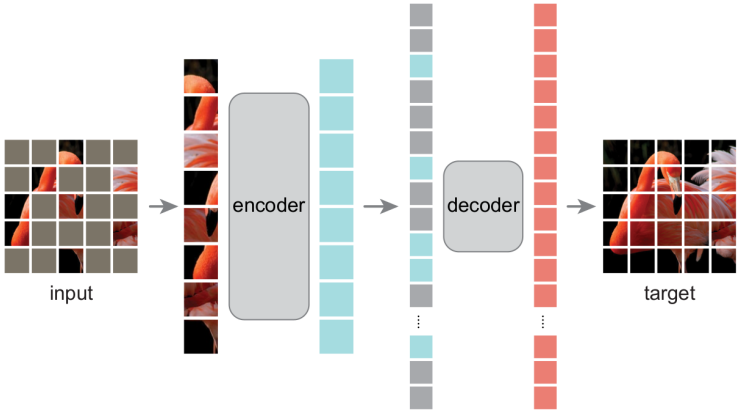

Masked LM




Next Sentence Prediction

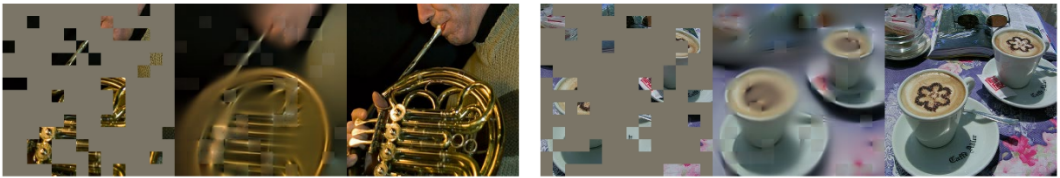
➤ Reconstruct the missing parts (pixel-level MSE loss)




➤ Vision Transformer architecture

➤ Classification performance (Top-1 accuracy) with ViT architectures, pre-trained and evaluated on ImageNet

| Approach | ViT-B | ViT-L | ViT-H | ViT-$H_{488}$ |
|---|---|---|---|---|
| Supervised | 82.3 | 82.6 | 83.1 | ✗ |
| MAE | 83.6 | 85.9 | 86.9 | **87.8** |



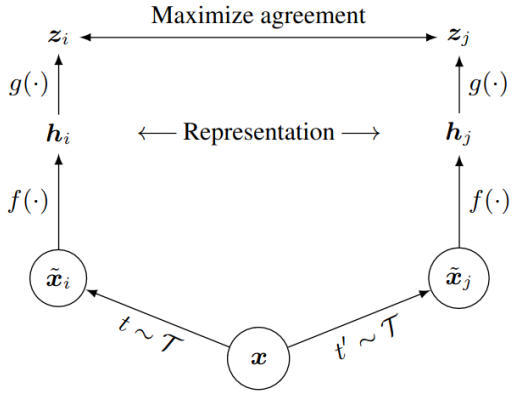
Left: masked input, Middle: reconstructed image, Right: ground truth

➤ Idea: generate two views for each input (= positive pair). Bring latent representations of positives closer and keep latent representations of negatives away

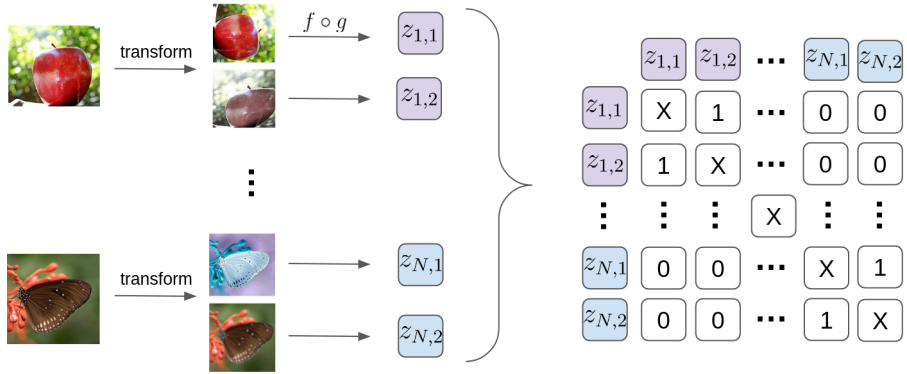
➤ Siamese networks: same architecture with same weights used on different inputs while comparing outputs

**Algorithm 1** SimCLR's main learning algorithm.

**input:** batch size $N$, constant $\tau$, structure of $f, g, \mathcal{T}$.
**for** sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  **for all** $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  **end for**
  **for all** $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  **end for**
  **define** $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks $f$ and $g$ to minimize $\mathcal{L}$
**end for**
**return** encoder network $f(\cdot)$, and throw away $g(\cdot)$



- Find the matching pairs
- Cosine similarity as distance metric




- Efficiency depends on mini-batch size

➤ Contrastive Language-Image Pre-training (CLIP)

**Idea**

Jointly train two encoders to avoid human annotation effort:

- An image encoder $f$: ResNet/ViT
- A text encoder $g$: transformer

➤ Both encoders are trained to generate a fixed-length latent representation from text/image input sharing the same feature space

**How?**

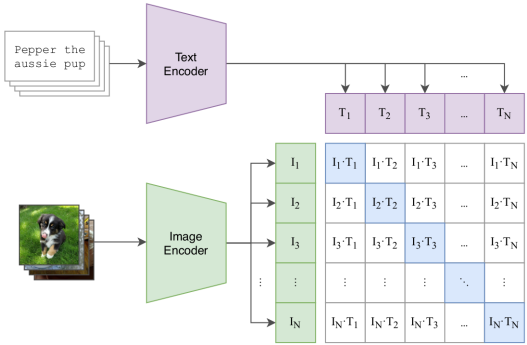
Constrative learning between images and captions

➤ 400 million pairs (image, text) collected from the web

$x$: image
$y$: caption

Image encoder: $f(x) = I$

Text encoder: $g(y) = T$




## Symmetric loss

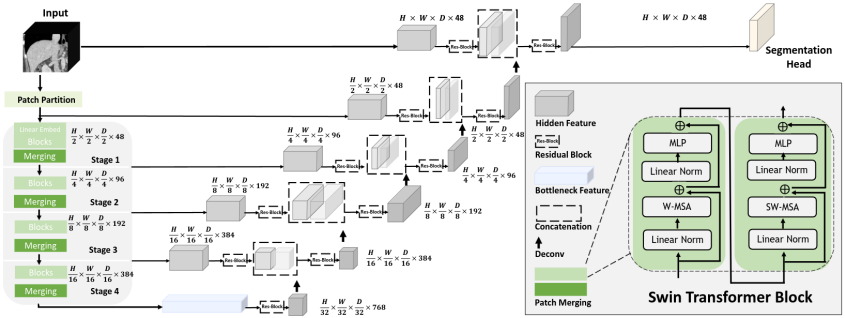
$$\mathcal{L} = \frac{1}{2}\Big(\sum_{i=1}^{N} \underbrace{\mathcal{L}_{\text{CE}}(\hat{y}_i^I, y_i^I)}_{\substack{\text{image } i \text{ compared} \\ \text{to all texts}}} + \sum_{t=1}^{N} \underbrace{\mathcal{L}_{\text{CE}}(\hat{y}_t^T, y_t^T)}_{\substack{\text{text } t \text{ compared} \\ \text{to all images}}}\Big)$$

$$\hat{y}_{i,j}^I = I_i \cdot T_j$$

$$\hat{y}_{t,j}^T = I_j \cdot T_t$$

# Plan du cours

➤ Medical image segmentation

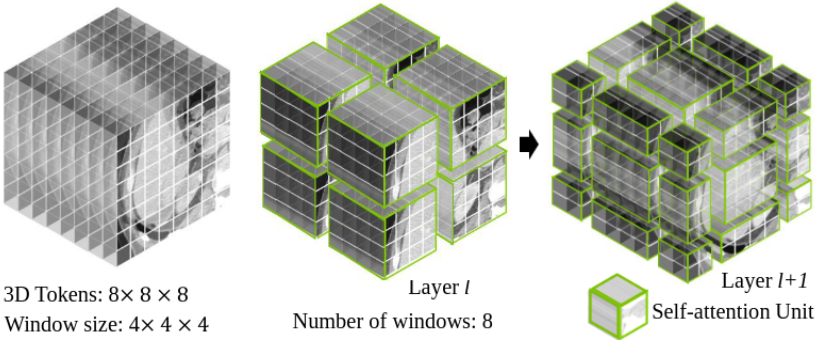



## Goal: reducing information compression when patching

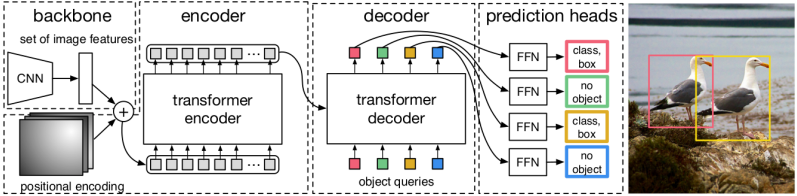
Size of input sub-volume: $128 \times 128 \times 128 \times 4$

Patch size: $2 \times 2 \times 2$

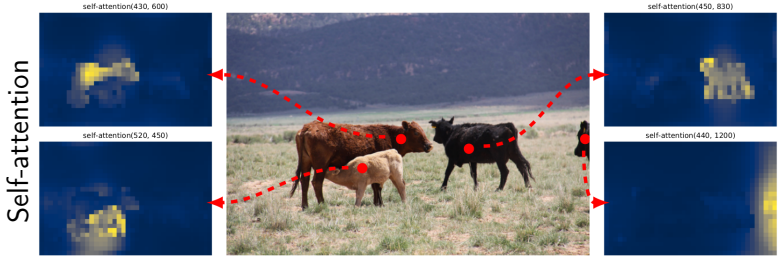
Input dimension for Swin: $(64 \times 64 \times 64) \times 48 \rightarrow 262,144 \times 48$

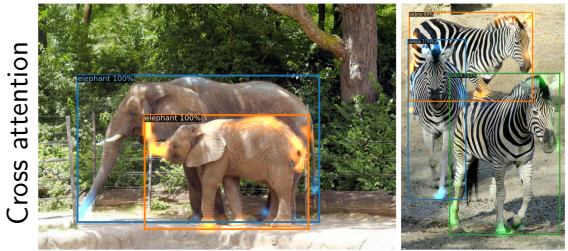

3D Tokens: 8× 8 × 8
Window size: 4× 4 × 4
Layer l
Number of windows: 8
Layer l+1
Self-attention Unit

➤ Object detection

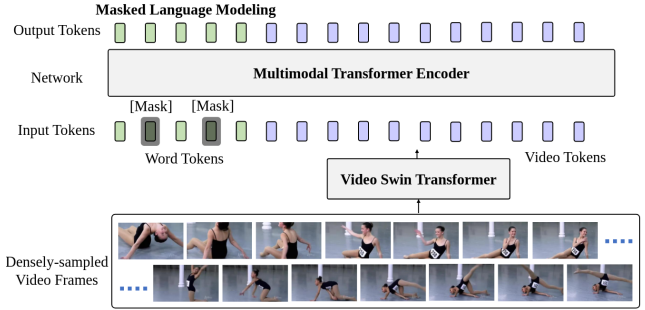



Set of $N$ learned object queries (must be high enough)
All predictions in parallel


Self-attention
self-attention(430, 600)
self-attention(520, 450)
self-attention(450, 830)
self-attention(440, 1200)


Cross attention
elephant 100%
elephant 100%

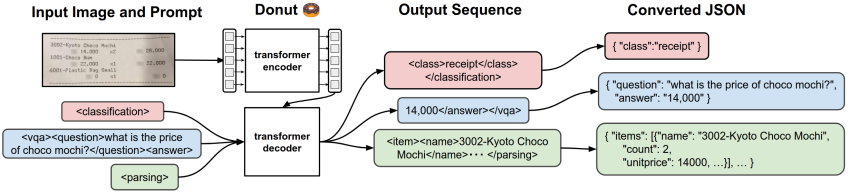
➤ Video captioning



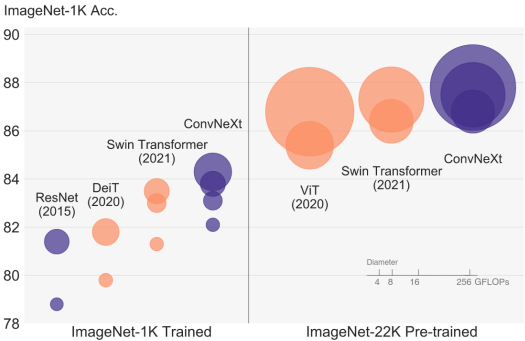

Generated caption: A dog is eating a watermelon
GT1: A dog is eating a watermelon
GT2: A dog eats watermelon

Generated caption: A boy is jumping on a trampoline
GT1: A boy jumps on a trampoline
GT2: Boy jumping on trampoline

➤ Document analysis and recognition



➤ ConvNext (2022) [14]




CNN with ViT/Swin patterns:

- ReLU $\rightarrow$ GELU
- Batch Norm $\rightarrow$ Layer Norm
- Less activations
- Larger kernel size
- etc

## Attention

- ✚ A powerful tool for feature extraction and encoder/decoder paradigm
- ━ Requires more data

## Transformer

- ✚ State of the art
- ✚ The same architecture for many modalities (text, image, audio)
- ✚ Parallelizable decoding at training time (teacher forcing)
- ━ Prohibitive for high resolution data (quadratic complexity)

# References I


[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations*. 2015.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 5998–6008.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021.

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *2021 IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9992–10002.

[5] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *14th European Conferenceon Computer Vision*. Vol. 9910. 2016, pp. 69–84.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186.

# References II


[7] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. "Masked Autoencoders Are Scalable Vision Learners". In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15979–15988.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. 2020, pp. 1597–1607.

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 8748–8763.

[10] Yucheng Tang, Dong Yang, Wenqi Li, Holger R. Roth, Bennett A. Landman, Daguang Xu, Vishwesh Nath, and Ali Hatamizadeh. "Self-Supervised Pre-Training of Swin Transformers for 3D Medical Image Analysis". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20698–20708.

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers". In: *16th European Conference on Computer Vision*. Vol. 12346. 2020, pp. 213–229.

# References III


[12] Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. "SwinBERT: End-to-End Transformers with Sparse Attention for Video Captioning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2022, pp. 17928–17937.

[13] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. "OCR-Free Document Understanding Transformer". In: *17th European Conference on Computer Vision*. Vol. 13688. 2022, pp. 498–517.

[14] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. "A ConvNet for the 2020s". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11966–11976.