

# Architecture de réseau de neurones - Le Transformer

Guillaume Bourmaud

# PLAN

I. Histoire du “Transformer”

II. Couche d’attention à softmax

III. Équivariance par permutation et encodage de la position

IV. Application à des images

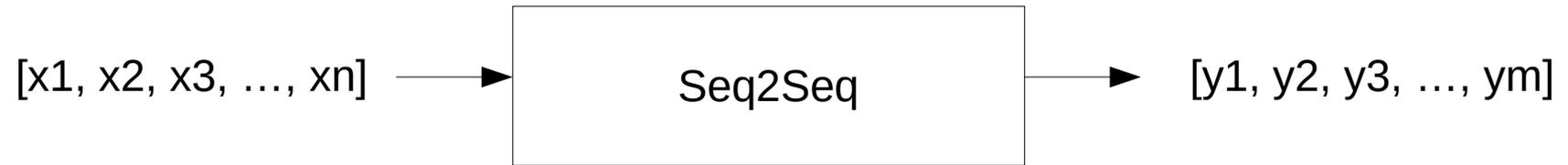
V. Limites

# I) Histoire du “Transformer”

“Attention is All You Need”, NIPS 2017

1)

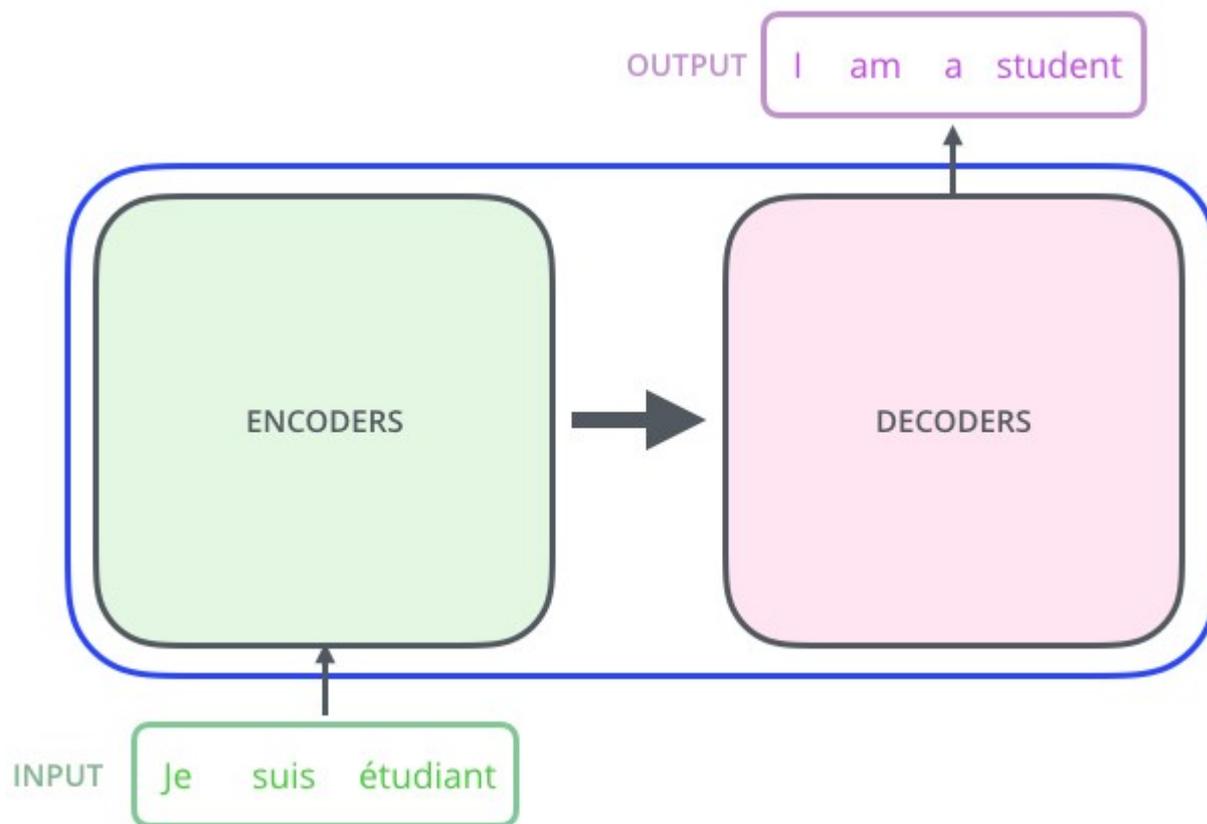
# Sequence-to-sequence



"the cat sat on the mat" -> [Seq2Seq model] -> "le chat etait assis sur le tapis"

1)

# Architecture encodeur-décodeur



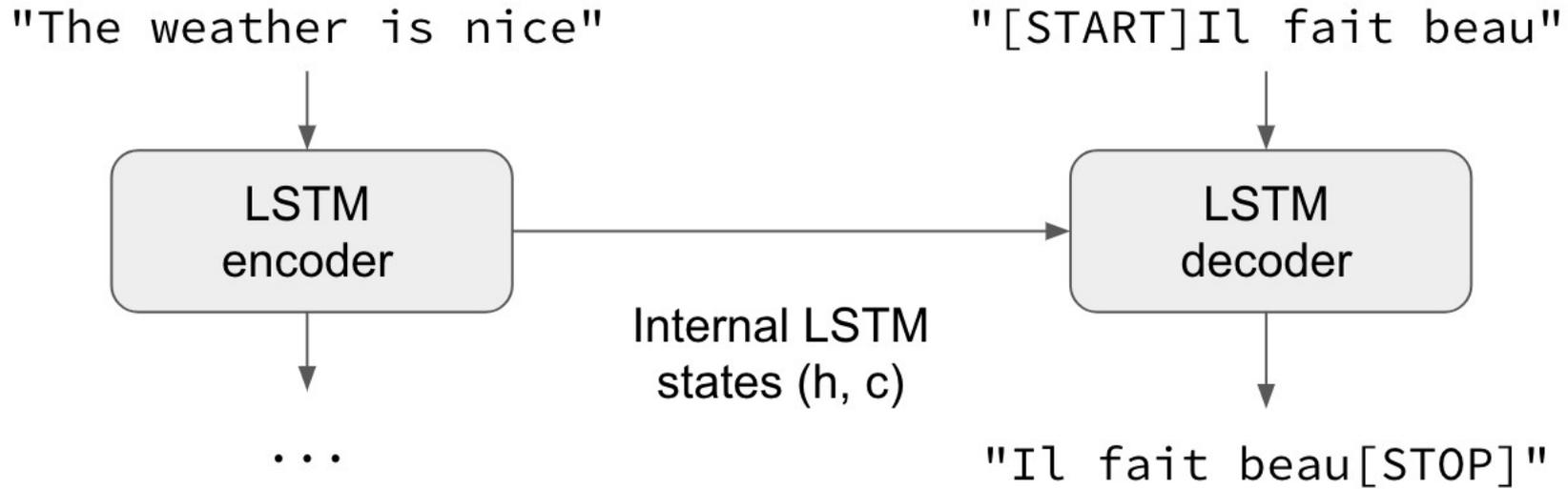
Encodeur :  
- RNN  
- ou CNN  
- ou Transformer

Décodeur :  
- RNN  
- ou CNN  
- ou Transformer

1)

# Exemple avec un RNN

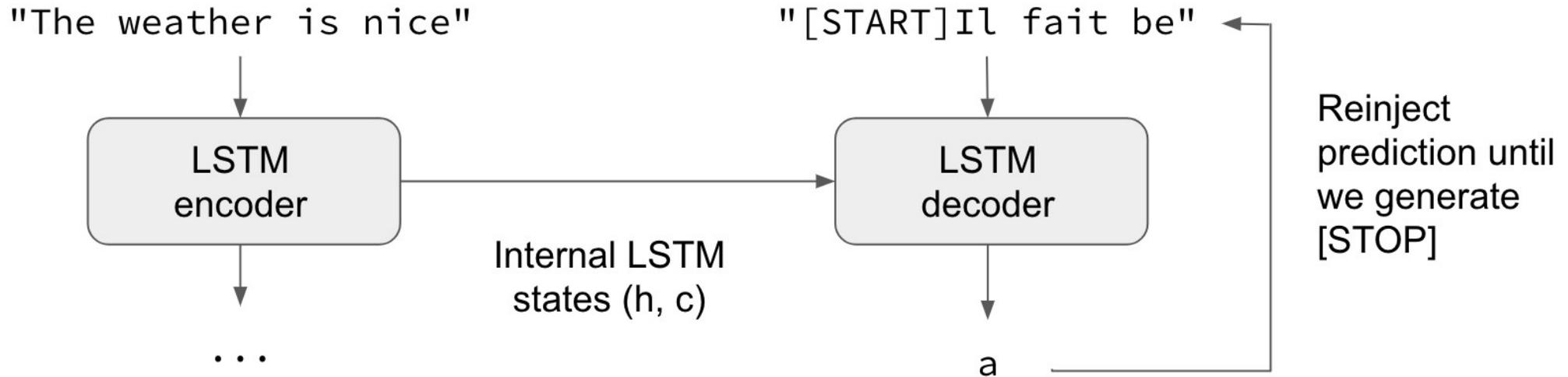
## Apprentissage (en "Teacher-Forcing")



1)

# Exemple avec un RNN (suite)

## Inférence

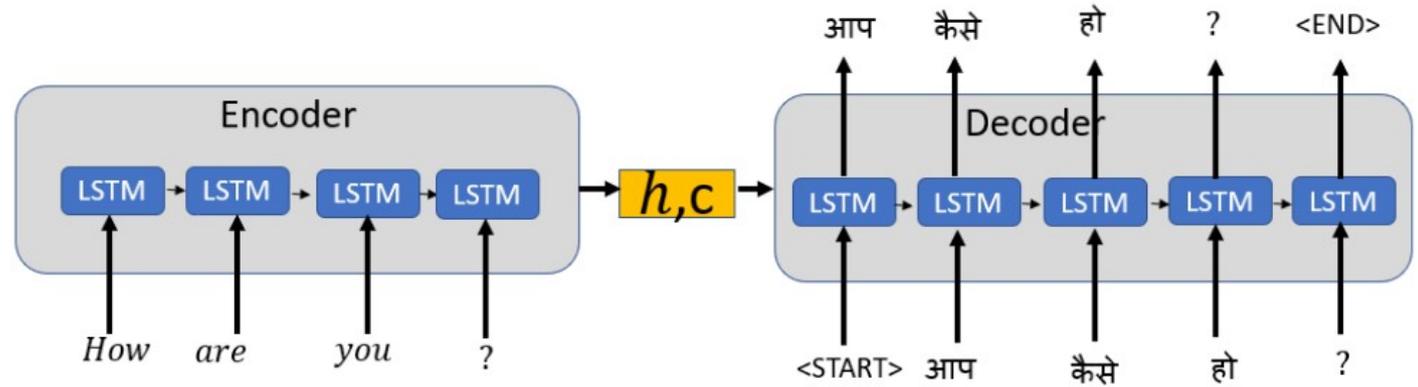


1)

# Exemple avec un RNN (suite)

## Limites

Calcul séquentiel  
↓  
Apprentissage lent  
↓  
Comment paralléliser ?



Difficile d'apprendre de longues dépendances avec un RNN.

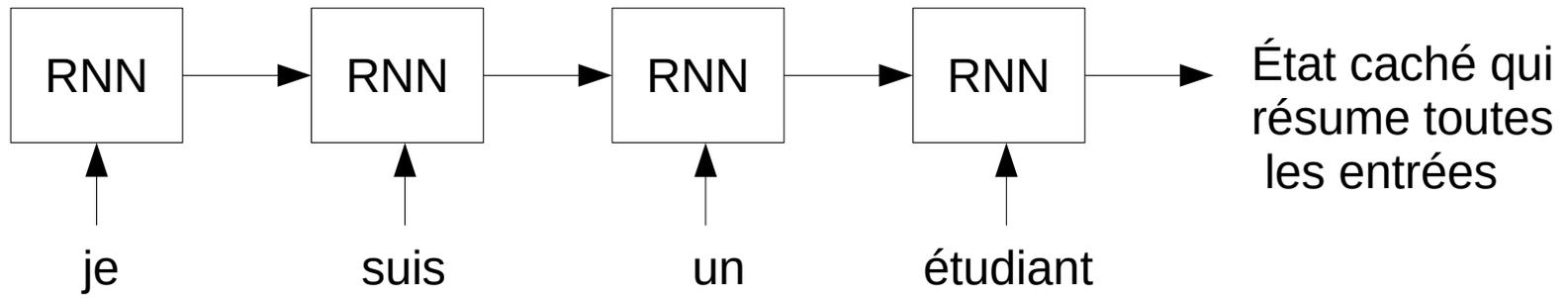
Coronavirus pandemic is spread across 175 countries, it is a serious problem especially in Italy, Spain and US as of March 2020.

1)

# Transformer vs RNN

## Encodeur RNN

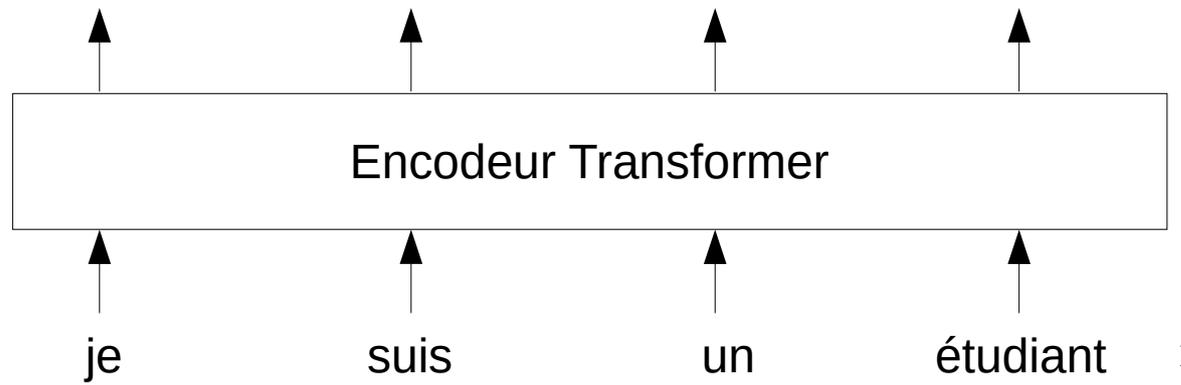
Calcul séquentiel  
+  
Chemin de la longueur  
de la phrase d'entrée



N entrées, N sorties. Chaque sortie dépend de **toutes** les entrées.

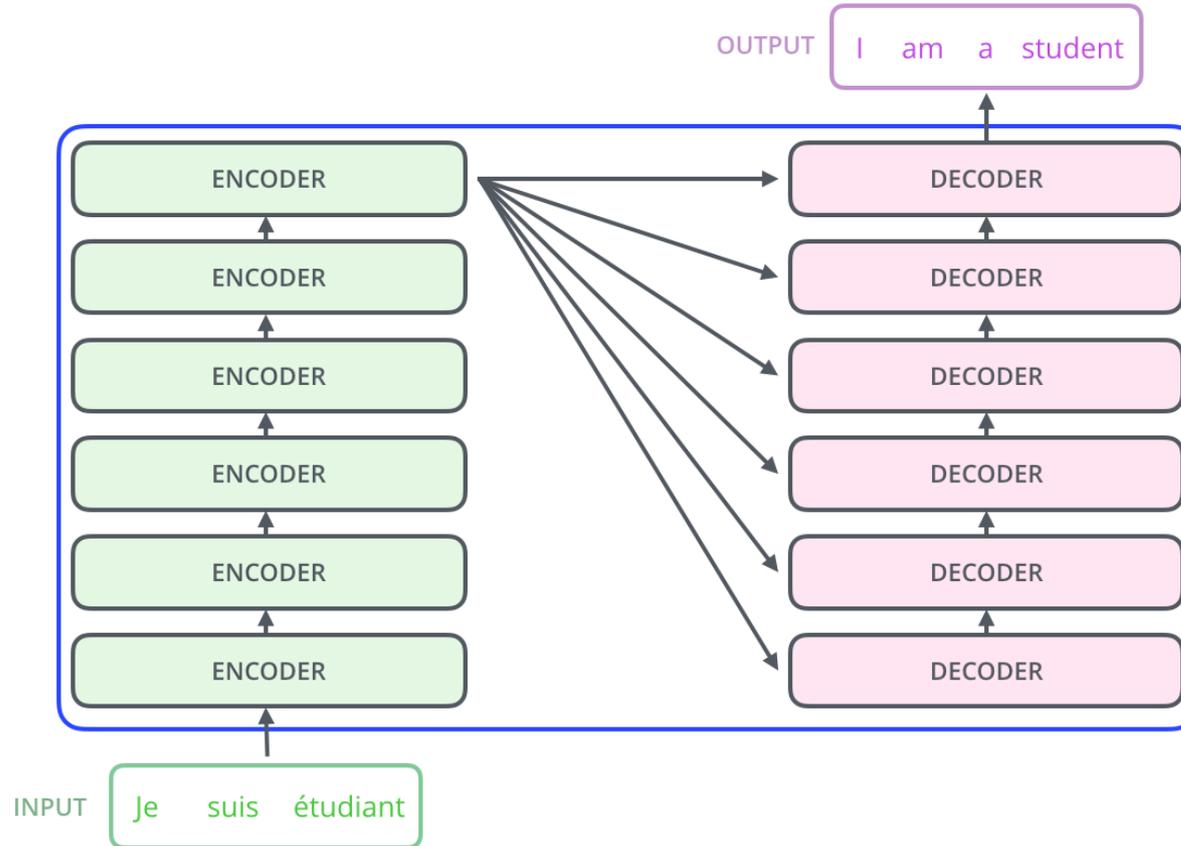
## Encodeur Transformer

Calcul en parallèle  
+  
Chemin de longueur  
constante



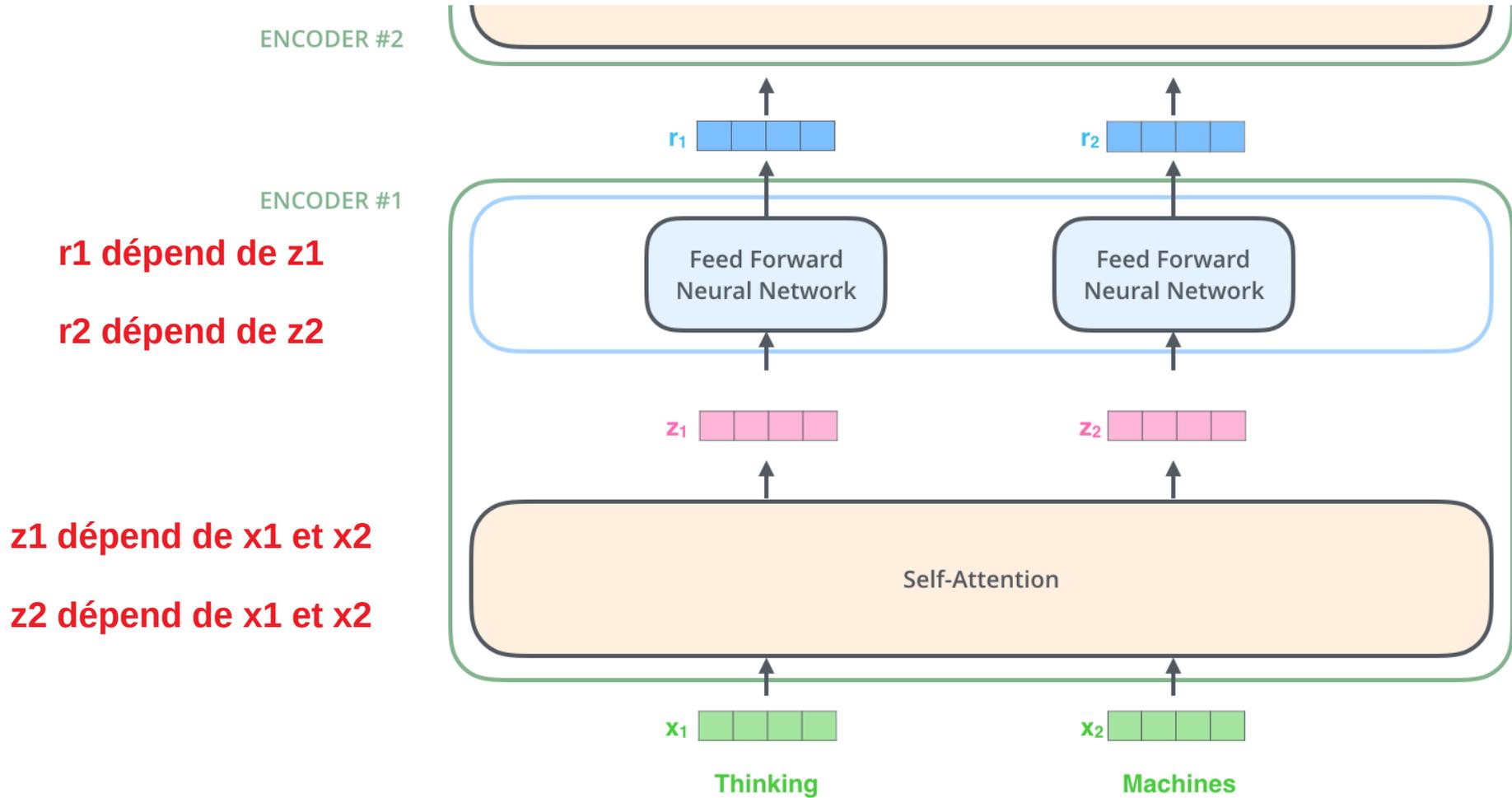
1)

# Transformer : vue globale



1)

# Transformer : encodeur



## II) Couche d'attention à softmax

II)

## Attention utilisant la fonction softmax

**Entrées**

$\mathbf{x}$  : vecteur de dimension  $1 \times D$

$\{\mathbf{y}_i\}_{i=1 \dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

II)

# Attention utilisant la fonction softmax

**Entrées**       $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
                   $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**       $\frac{\exp(\mathbf{x}\mathbf{y}_j^\top)}{\sum_i \exp(\mathbf{x}\mathbf{y}_i^\top)}$

Produit scalaire + exp :

- $\gg 1$  si  $\mathbf{x}$  est “attiré” par  $\mathbf{y}_j$
- $= 1$  si  $\mathbf{x}$  orthogonal à  $\mathbf{y}_j$
- $\approx 0$  si  $\mathbf{x}$  est “repoussé” par  $\mathbf{y}_j$

II)

# Attention utilisant la fonction softmax

**Entrées**       $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
                   $\{\mathbf{y}_i\}_{i=1 \dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**       $\frac{\exp(\mathbf{x}\mathbf{y}_j^\top)}{\sum_i \exp(\mathbf{x}\mathbf{y}_i^\top)}$

Produit scalaire + exp :

$\gg 1$  si  $\mathbf{X}$  est "attiré" par  $\mathbf{Y}_j$    $\mathbf{Y}_j$  "attire l'attention de"  $\mathbf{X}$   
 $= 1$  si  $\mathbf{X}$  orthogonal à  $\mathbf{Y}_j$   
 $\approx 0$  si  $\mathbf{X}$  est "repoussé" par  $\mathbf{Y}_j$

II)

# Attention utilisant la fonction softmax

**Entrées**

$\mathbf{x}$  : vecteur de dimension  $1 \times D$

$\{\mathbf{y}_i\}_{i=1 \dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**

$$\frac{\exp(\mathbf{x}\mathbf{y}_j^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}\mathbf{y}_k^\top)} \leftarrow \text{Softmax}$$

II)

# Attention utilisant la fonction softmax

**Entrées**       $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
                    $\{\mathbf{y}_i\}_{i=1 \dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**      
$$\mathbf{x}' = \sum_{j=1}^{N_y} \frac{\exp(\mathbf{x}\mathbf{y}_j^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}\mathbf{y}_k^\top)} \mathbf{y}_j$$

Combinaison linéaire des  $\{\mathbf{y}_i\}_{i=1 \dots N_y}$

Les poids les plus élevés de cette combinaison linéaire correspondent aux vecteurs ayant le plus “attiré l’attention” de  $\mathbf{X}$

II)

# Attention utilisant la fonction softmax

**Entrées**       $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
                    $\{\mathbf{y}_i\}_{i=1 \dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**

$$\mathbf{x}' = \sum_{j=1}^{N_y} \frac{\exp(\mathbf{x}Q(\mathbf{y}_jK)^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}Q(\mathbf{y}_kK)^\top)} \mathbf{y}_j$$

Pour pouvoir apprendre à “attirer l’attention” → introduction de paramètres à optimiser

**Paramètres**       $Q$  : matrice “query” de taille  $D \times L$   
                            $K$  : matrice “key” de taille  $D \times L$

II)

# Attention utilisant la fonction softmax

**Entrées**       $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
                    $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**      
$$\mathbf{x}' = \mathbf{x} + \sum_{j=1}^{N_y} \frac{\exp(\mathbf{x}Q(\mathbf{y}_jK)^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}Q(\mathbf{y}_kK)^\top)} \mathbf{y}_jV$$

En pratique, la combinaison linéaire est elle-même transformée linéairement, et suivie d'une connection résiduelle.

**Paramètres**       $Q$  : matrice "query" de taille  $D \times L$   
                            $K$  : matrice "key" de taille  $D \times L$   
                            $V$  : matrice "value" de taille  $D \times D$

# Attention utilisant la fonction softmax

**Entrées**

$\mathbf{x}$  : vecteur de dimension  $1 \times D$

$\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction**

$$\mathbf{x}' = \mathbf{x} + \sum_{j=1}^{N_y} \frac{\exp(\mathbf{xQ}(\mathbf{y}_j\mathbf{K})^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{xQ}(\mathbf{y}_k\mathbf{K})^\top)} \mathbf{y}_j\mathbf{V}$$


---

**Paramètres**

$\mathbf{Q}$  : matrice “query” de taille  $D \times L$

$\mathbf{K}$  : matrice “key” de taille  $D \times L$

$\mathbf{V}$  : matrice “value” de taille  $D \times D$

II)

## Couche d'inter-attention softmax ("Cross-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X$  : matrice de taille  $N_x \times D$   
 $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : vecteurs de dimension  $D$   $\longrightarrow$   $Y$  : matrice de taille  $N_y \times D$

---

II)

## Couche d'inter-attention softmax ("Cross-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X$  : matrice de taille  $N_x \times D$   
 $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : vecteurs de dimension  $D$   $\longrightarrow$   $Y$  : matrice de taille  $N_y \times D$

---

**Sorties**  $\{\mathbf{x}'_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X'$  : matrice de taille  $N_x \times D$

II)

## Couche d'inter-attention softmax ("Cross-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X$  : matrice de taille  $N_x \times D$   
 $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : vecteurs de dimension  $D$   $\longrightarrow$   $Y$  : matrice de taille  $N_y \times D$

---

**Sorties**  $\{\mathbf{x}'_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X'$  : matrice de taille  $N_x \times D$

---

$$M = XQ(YK)^\top$$

$$S = \text{softmax}(M, \text{dim}=1)$$

$$X' = X + SYV$$

Calcul réalisé en parallèle sur  
les vecteurs  $\{\mathbf{x}_i\}_{i=1\dots N_x}$

II)

## Couche d'inter-attention softmax ("Cross-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X$  : matrice de taille  $N_x \times D$   
 $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : vecteurs de dimension  $D$   $\longrightarrow$   $Y$  : matrice de taille  $N_y \times D$

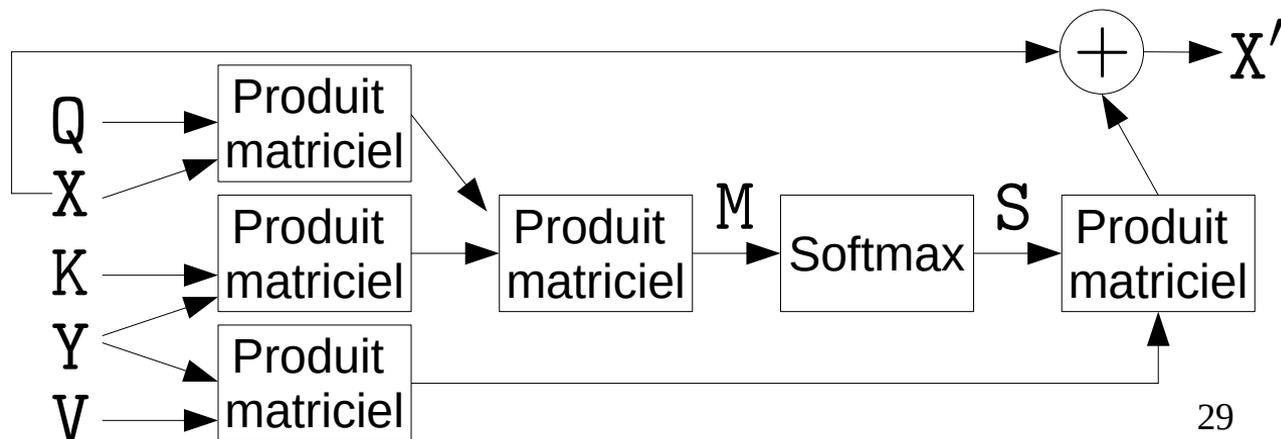
**Sorties**  $\{\mathbf{x}'_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $X'$  : matrice de taille  $N_x \times D$

$$M = XQ(YK)^\top$$

$$S = \text{softmax}(M, \text{dim}=1)$$

$$X' = X + SYV$$

Calcul réalisé en parallèle sur  
les vecteurs  $\{\mathbf{x}_i\}_{i=1\dots N_x}$



II)

## Couche d'inter-attention softmax ("Cross-attention") (suite)

$$\mathbf{X} : N_x \times D \quad \mathbf{Y} : N_y \times D \quad \mathbf{X}' : N_x \times D$$

---

$$\mathbf{M} = \mathbf{XQ}(\mathbf{YK})^\top : N_x \times N_y \quad \mathbf{S} = \text{softmax}(\mathbf{M}, \text{dim}=1) : N_x \times N_y$$

II)

## Couche d'inter-attention softmax ("Cross-attention") (suite)

$$X : N_x \times D \quad Y : N_y \times D \quad X' : N_x \times D$$

---

$$M = XQ(YK)^\top : N_x \times N_y$$

$$S = \text{softmax}(M, \text{dim}=1) : N_x \times N_y$$

### Calcul

- Nombre d'opérations  
potentiellement très élevé

+ Parallélisable

### Stockage

- Mémoire requise pour stocker M et S  
potentiellement très élevée

II)

# Couche d'inter-attention softmax ("Cross-attention") (suite)

$$X : N_x \times D \quad Y : N_y \times D \quad X' : N_x \times D$$

$$M = XQ(YK)^\top : N_x \times N_y \quad S = \text{softmax}(M, \text{dim}=1) : N_x \times N_y$$

## Calcul

- Nombre d'opérations  
potentiellement très élevé

+ Parallélisable

## Stockage

- Mémoire requise pour stocker M et S  
potentiellement très élevée

Exemple :  $N_x = N_y = 640 \times 480 \approx 3.10^5$  pixels  
 $N_x \times N_y \approx 9.10^{10}$  flottants (32 bits)  $\rightarrow$  360Go

II)

## Cas particulier : Couche d'auto-attention ("Self-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $\mathbf{X}$  : matrice de taille  $N_x \times D$

---

**Sorties**  $\{\mathbf{x}'_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $\mathbf{X}'$  : matrice de taille  $N_x \times D$

---

II)

## Cas particulier : Couche d'auto-attention ("Self-attention")

**Entrées**  $\{\mathbf{x}_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $\mathbf{X}$  : matrice de taille  $N_x \times D$

---

**Sorties**  $\{\mathbf{x}'_i\}_{i=1\dots N_x}$  : vecteurs de dimension  $D$   $\longrightarrow$   $\mathbf{X}'$  : matrice de taille  $N_x \times D$

---

$$\mathbf{M} = \mathbf{XQ}(\mathbf{XK})^\top : N_x \times N_x$$

$$\mathbf{S} = \text{softmax}(\mathbf{M}, \text{dim}=1) : N_x \times N_x$$

$$\mathbf{X}' = \mathbf{X} + \mathbf{SXV} : N_x \times D$$

Transfert d'information depuis  $\mathbf{X}$  vers lui-même le tout stocké dans  $\mathbf{X}'$

II)

# Attention softmax à têtes multiples

## ”Multi-head dot-product attention”

**Entrées**

$\mathbf{x}$  : vecteur de dimension  $1 \times D$

$\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

1 tête = 1 façon  
”d’attirer l’attention”

$$\mathbf{x}' = \mathbf{x} + \sum_{j=1}^{N_y} \frac{\exp(\mathbf{x}Q(\mathbf{y}_jK)^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}Q(\mathbf{y}_kK)^\top)} \mathbf{y}_jV$$

II)

# Attention softmax à têtes multiples

## ”Multi-head dot-product attention”

Entrées

$\mathbf{x}$  : vecteur de dimension  $1 \times D$

$\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

1 tête = 1 façon  
”d’attirer l’attention”

$$\mathbf{x}' = \mathbf{x} + \sum_{j=1}^{N_y} \frac{\exp(\mathbf{xQ}(\mathbf{y}_j\mathbf{K})^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{xQ}(\mathbf{y}_k\mathbf{K})^\top)} \mathbf{y}_j \mathbf{V}$$

H têtes = H façons  
”d’attirer l’attention”

$$\mathbf{x}' = \mathbf{x} + \sum_{h=1}^H \sum_{j=1}^{N_y} \frac{\exp(\mathbf{xQ}_h(\mathbf{y}_j\mathbf{K}_h)^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{xQ}_h(\mathbf{y}_k\mathbf{K}_h)^\top)} \mathbf{y}_j \mathbf{V}_h$$

II)

# Attention softmax à têtes multiples (suite)

## ”Multi-head dot-product attention”

**Entrées**  $\mathbf{x}$  : vecteur de dimension  $1 \times D$   
 $\{\mathbf{y}_i\}_{i=1\dots N_y}$  : ensemble de vecteurs de dimension  $1 \times D$

---

**Fonction** 
$$\mathbf{x}' = \mathbf{x} + \sum_{h=1}^H \sum_{j=1}^{N_y} \frac{\exp(\mathbf{x} \mathbf{Q}_h (\mathbf{y}_j \mathbf{K}_h)^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x} \mathbf{Q}_h (\mathbf{y}_k \mathbf{K}_h)^\top)} \mathbf{y}_j \mathbf{V}_h \mathbf{W}_h$$

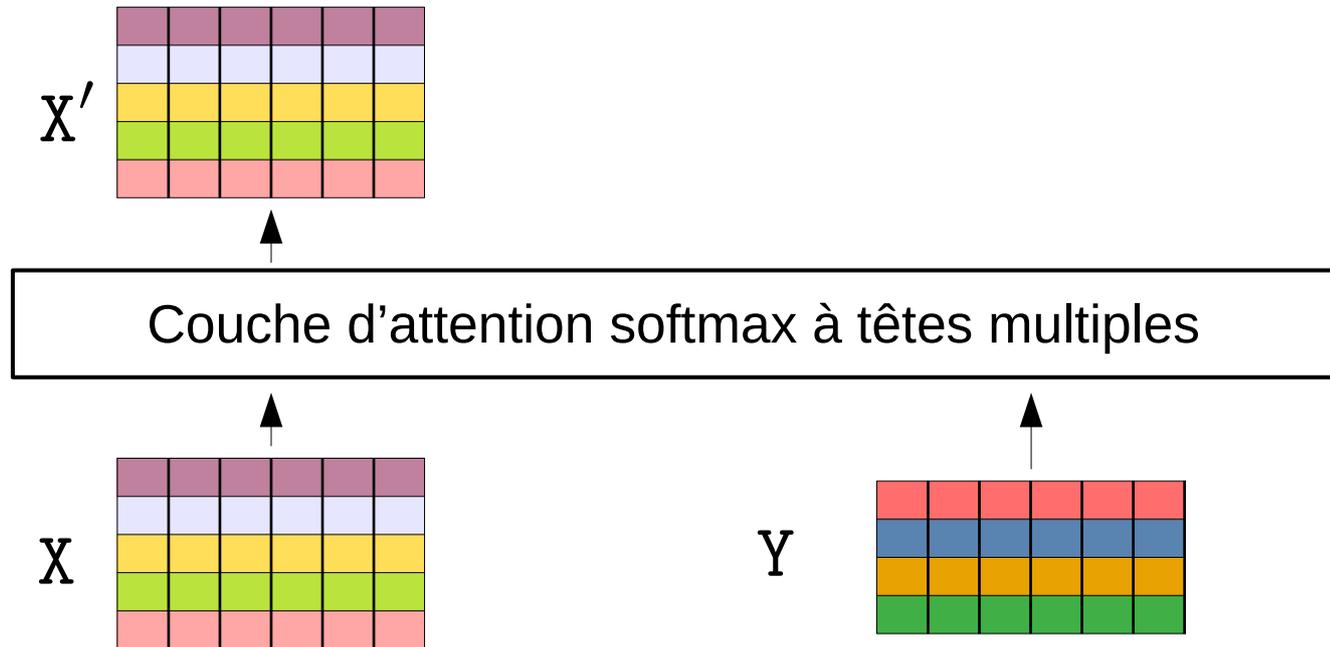
---

**Paramètres**  $\{\mathbf{Q}_h\}_{h=1\dots H}$  : matrices “query” de taille  $D \times L$   
 $\{\mathbf{K}_h\}_{h=1\dots H}$  : matrices “key” de taille  $D \times L$   
 $\{\mathbf{V}_h\}_{h=1\dots H}$  : matrices “value” de taille  $D \times L$   
 $\{\mathbf{W}_h\}_{h=1\dots H}$  : matrices “output” de taille  $L \times D$

II)

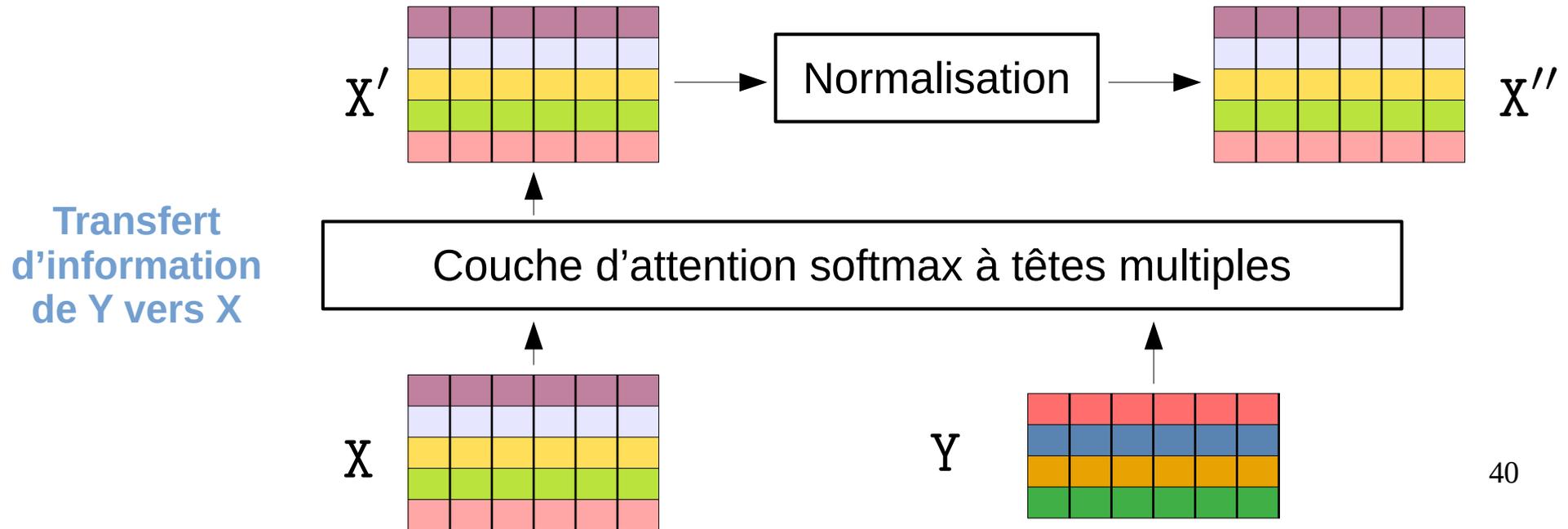
# Bloc d'attention "classique"

Transfert  
d'information  
de Y vers X



II)

# Bloc d'attention "classique"

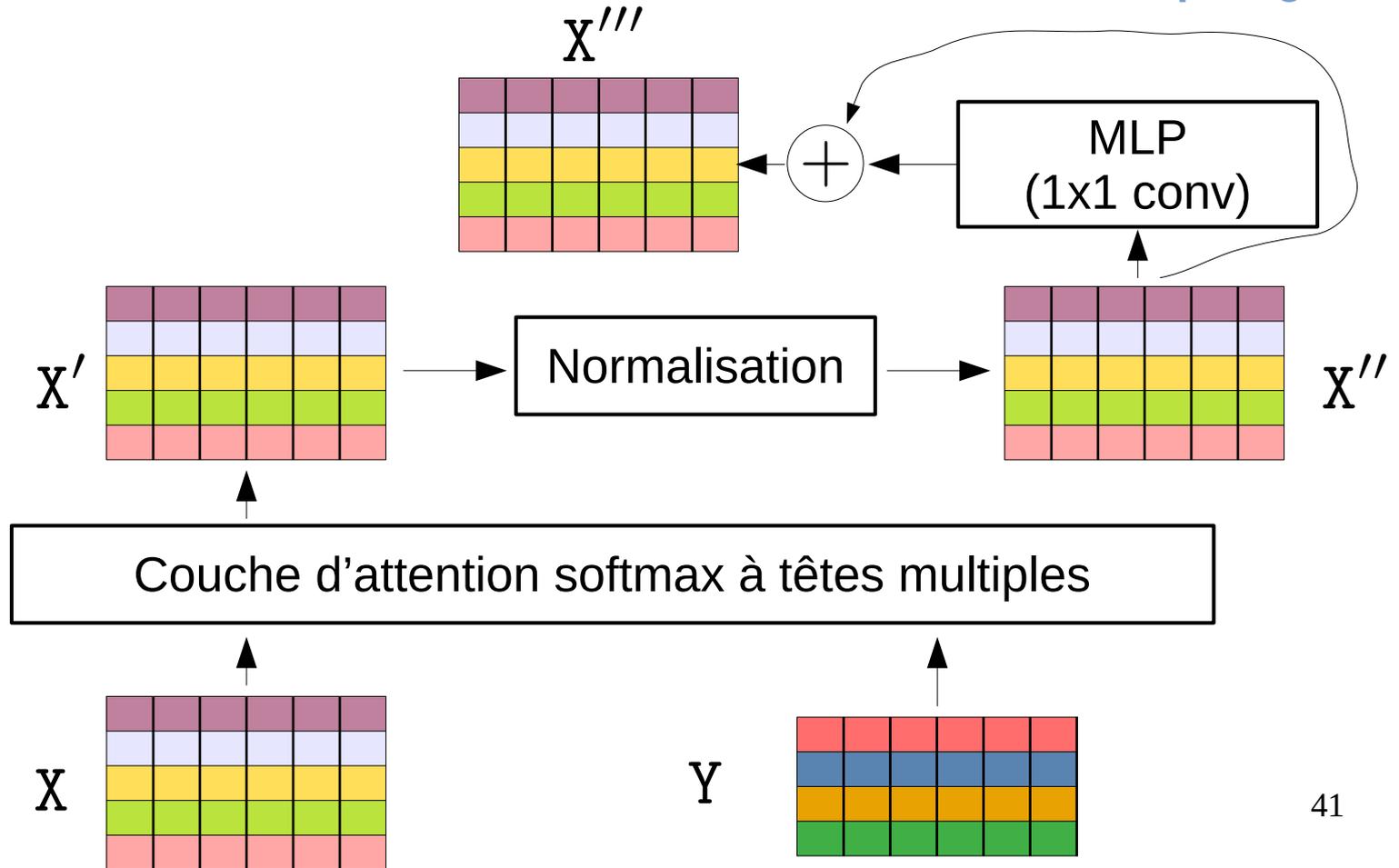


II)

# Bloc d'attention "classique"

Transformation non-linéaire de chaque ligne

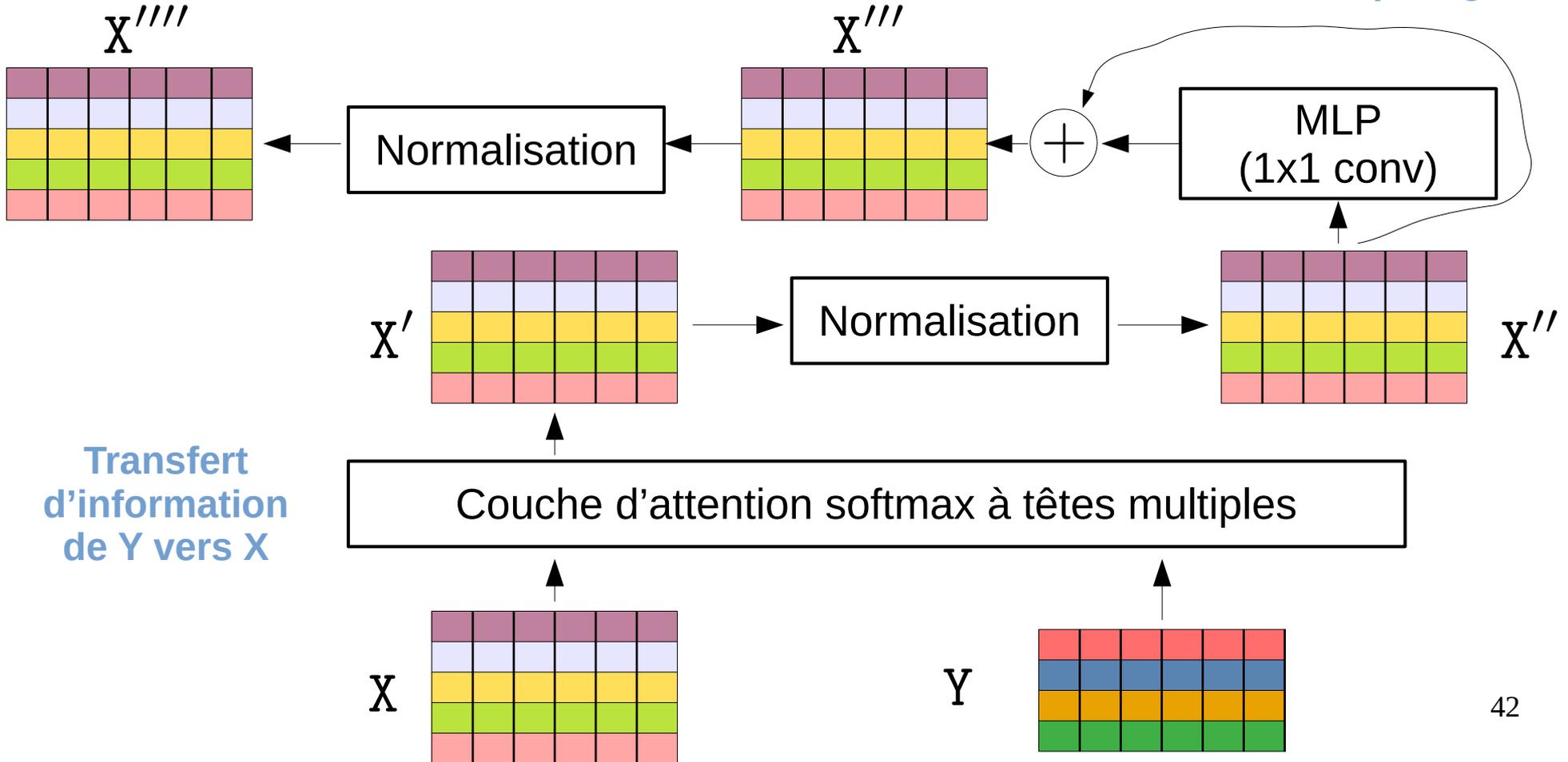
Transfert d'information de Y vers X



II)

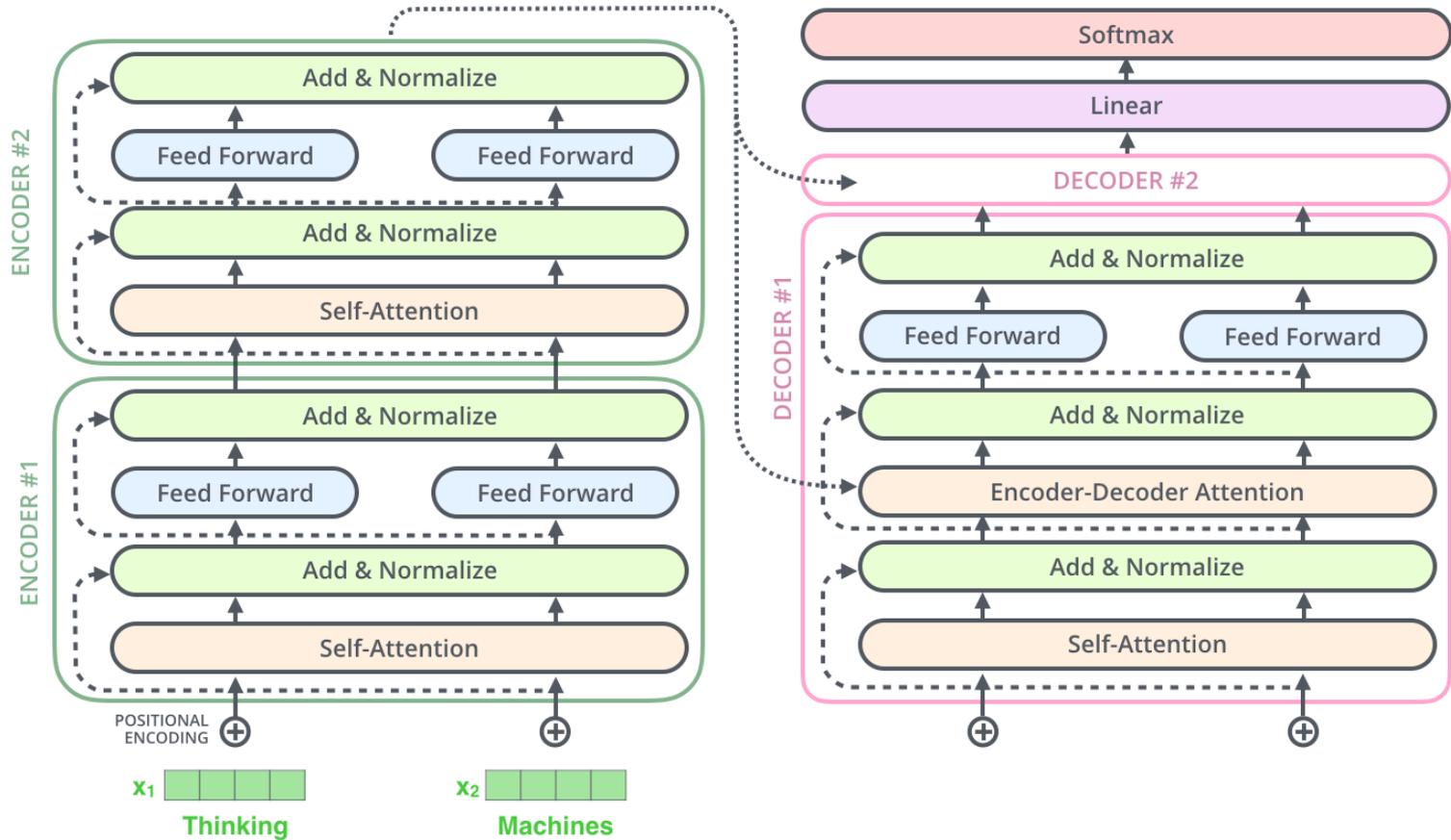
# Bloc d'attention "classique"

Transformation non-linéaire de chaque ligne



II)

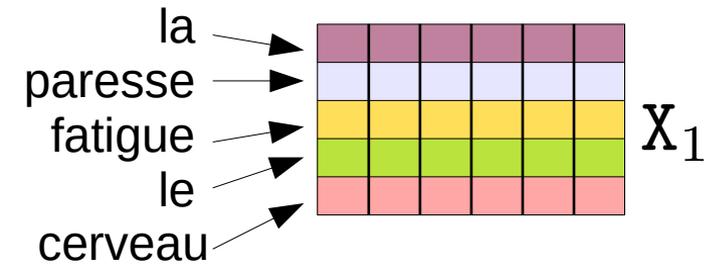
# Vue détaillée du Transformer



### III) Équivariance par permutation et encodage de la position

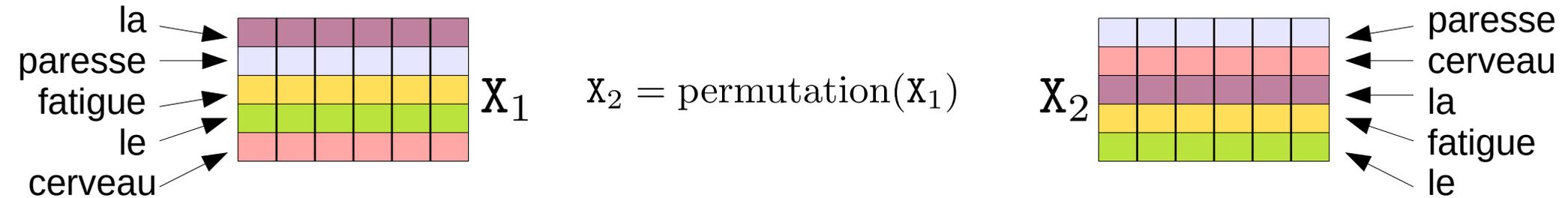
III)

# Non-équivalence par permutation



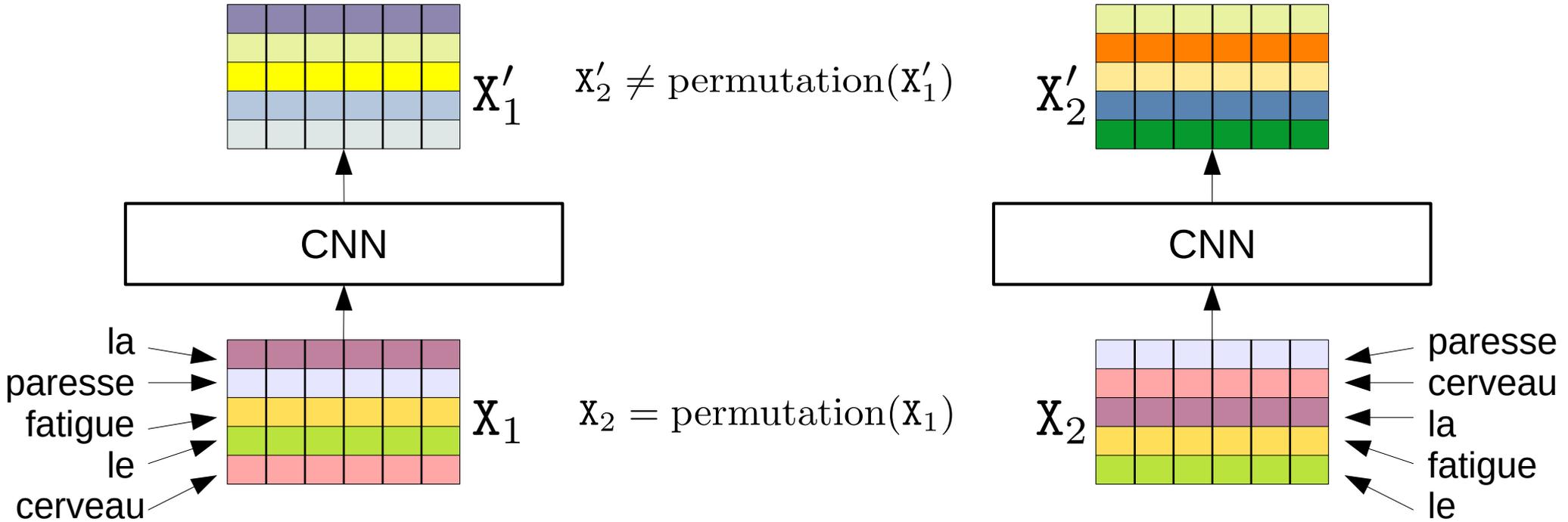
III)

# Non-équivalence par permutation



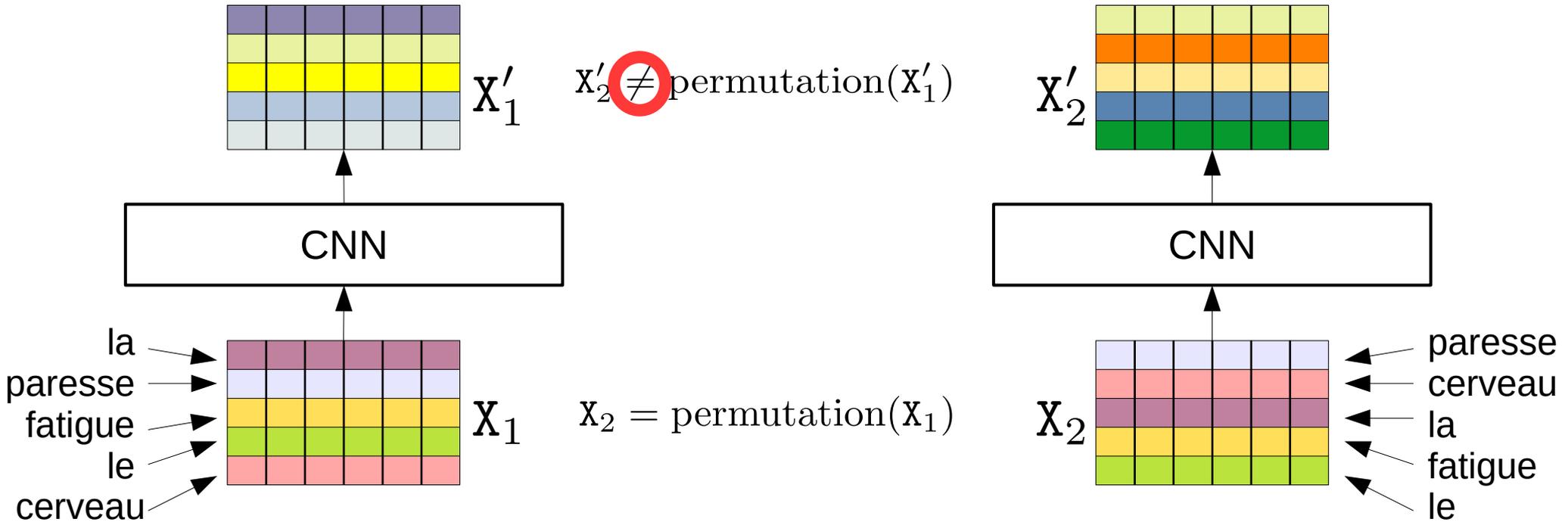
III)

# Non-équivalence par permutation



III)

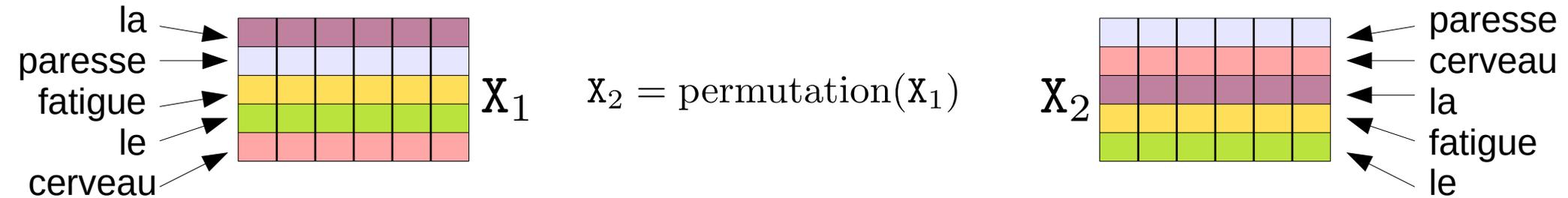
# Non-équivalence par permutation



CNN adapté aux ensembles ordonnés (phrase, signal, image, etc.)  
→ ses filtres s'appliquent sur un voisinage

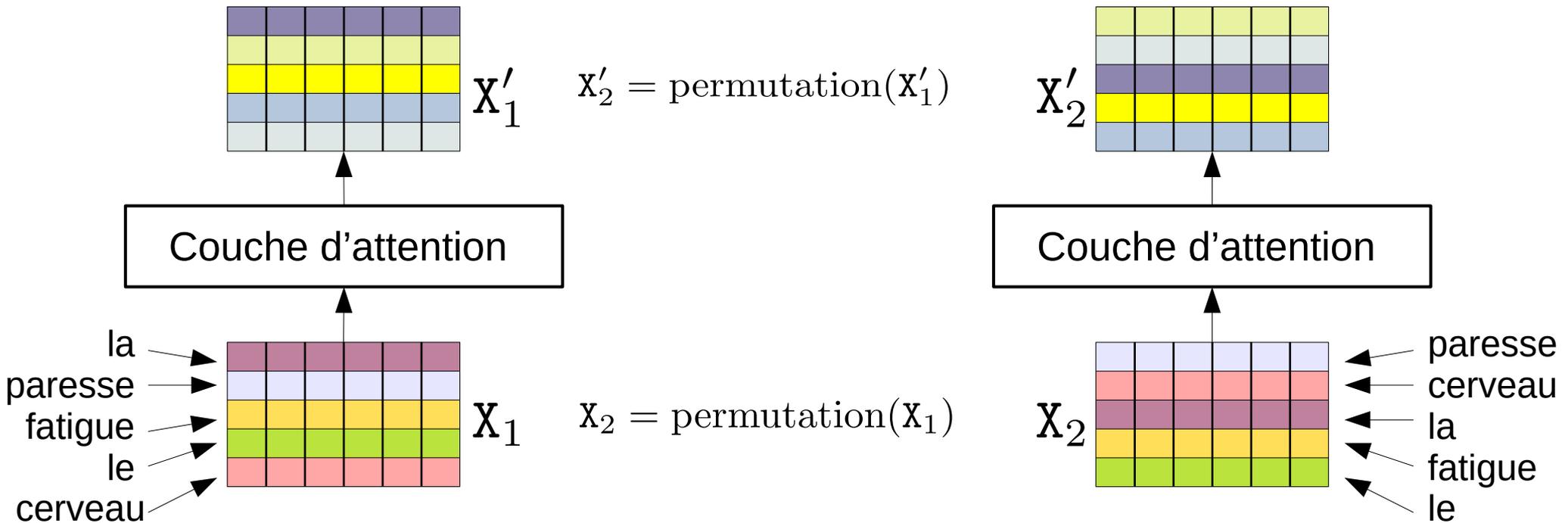
III)

# Équivariance par permutation



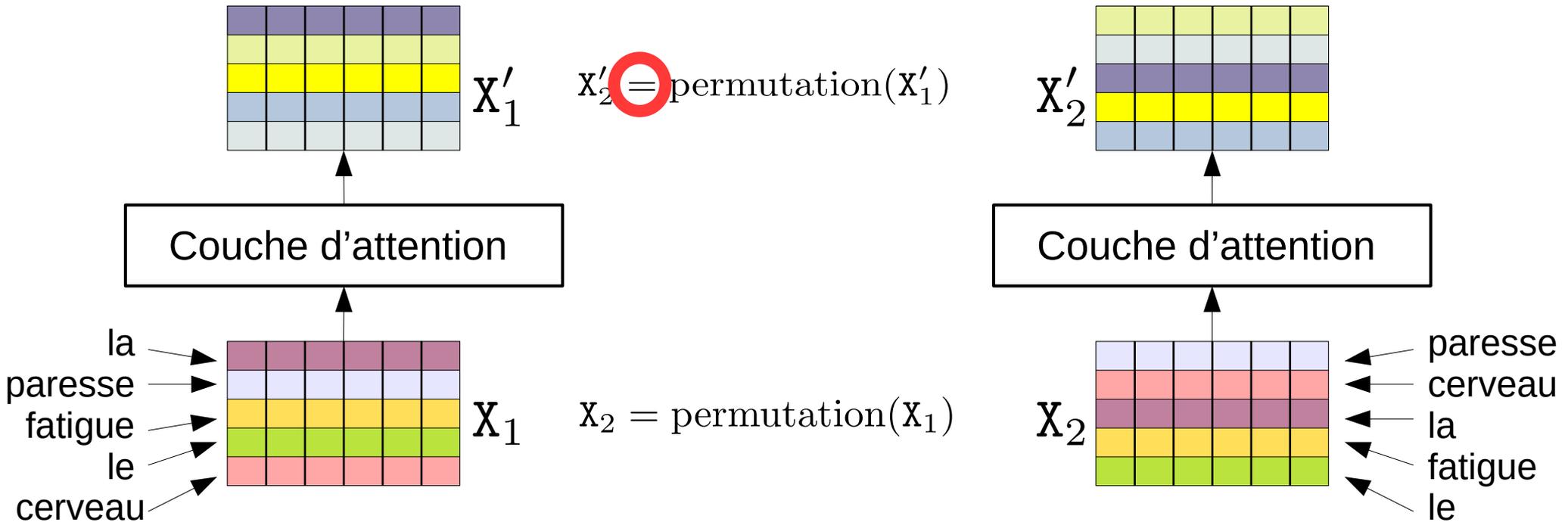
III)

# Équivariance par permutation



III)

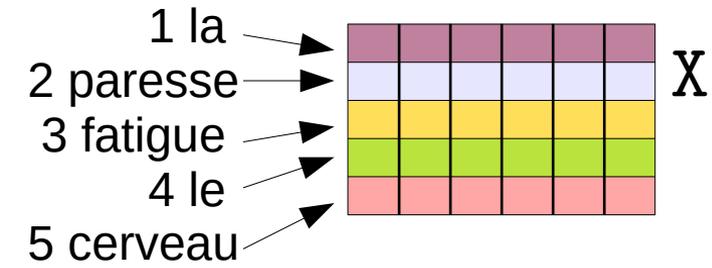
# Équivariance par permutation



**Pour un ensemble ordonné (phrase, signal, image, etc.)  
→ nécessité d'encoder la position**

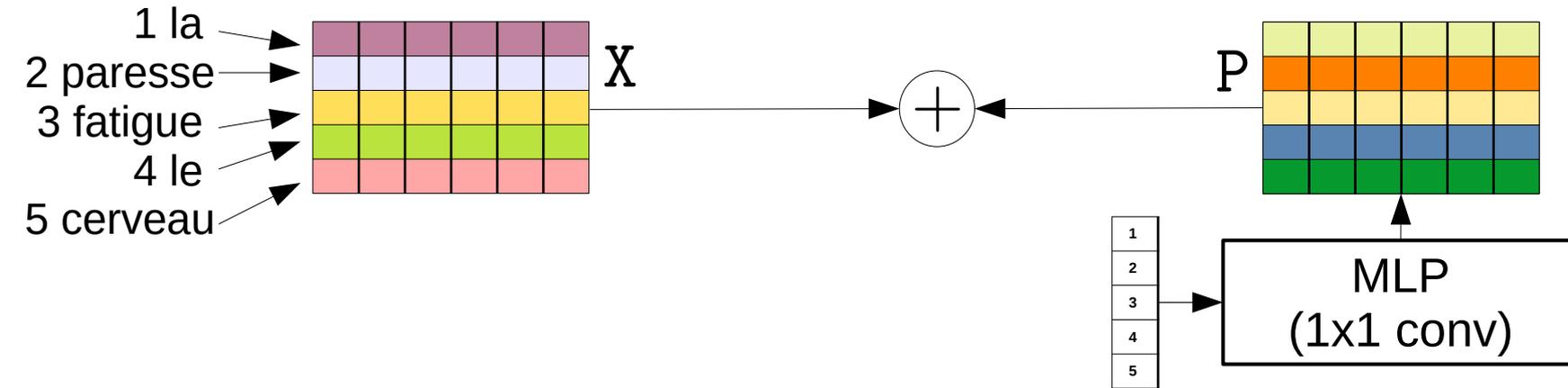
III)

# Encodage de la position



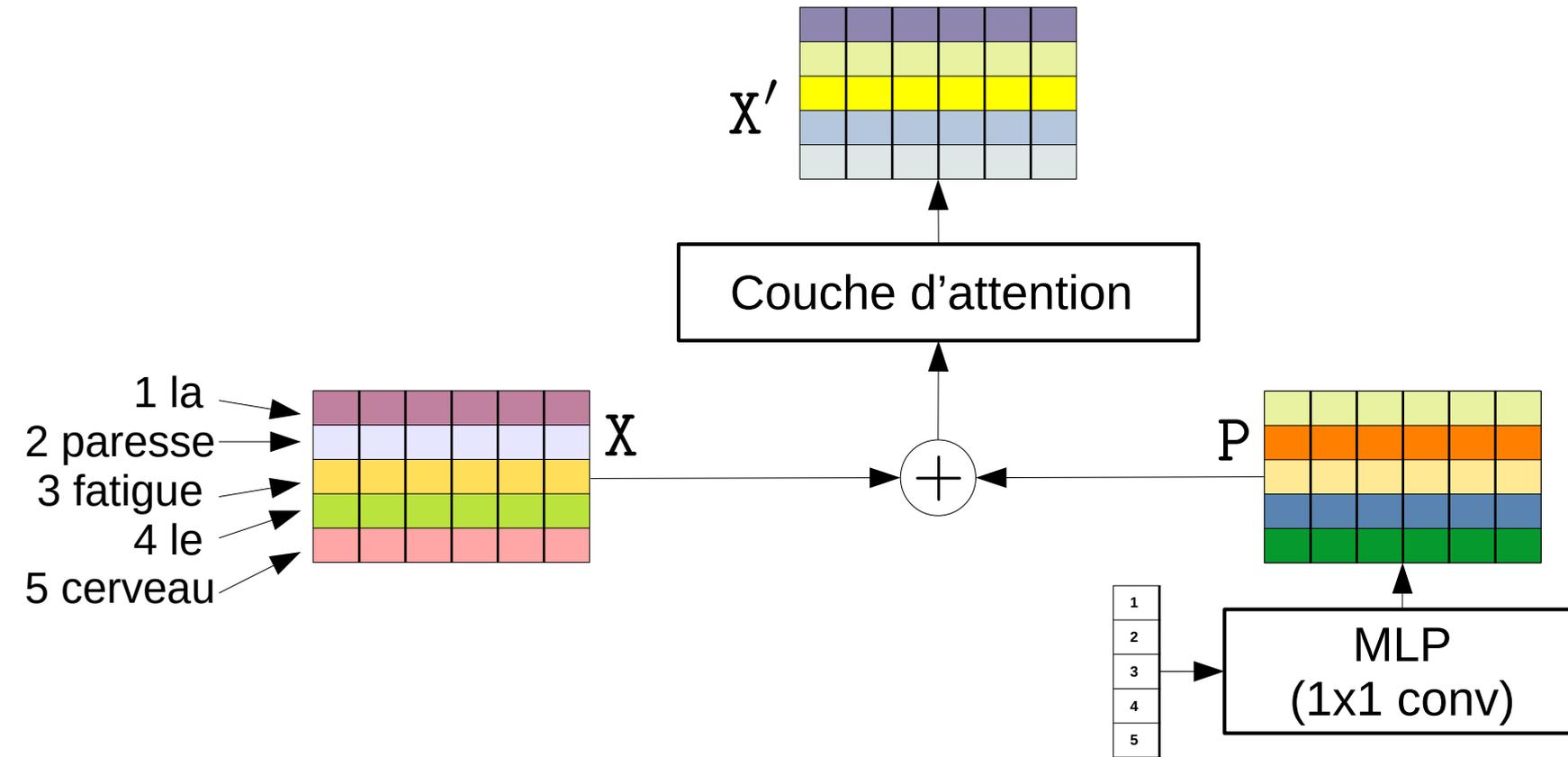
III)

# Encodage de la position



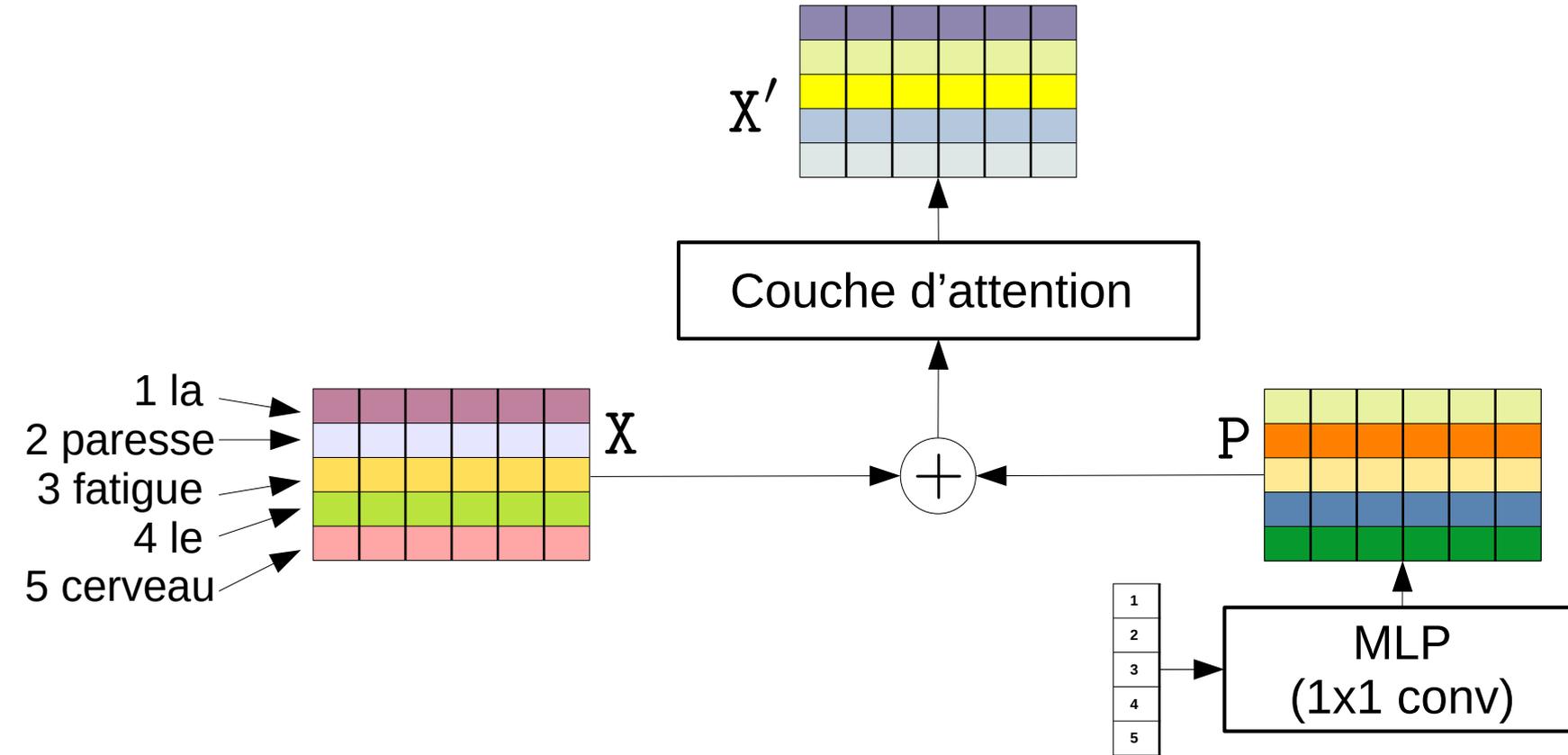
III)

# Encodage de la position



III)

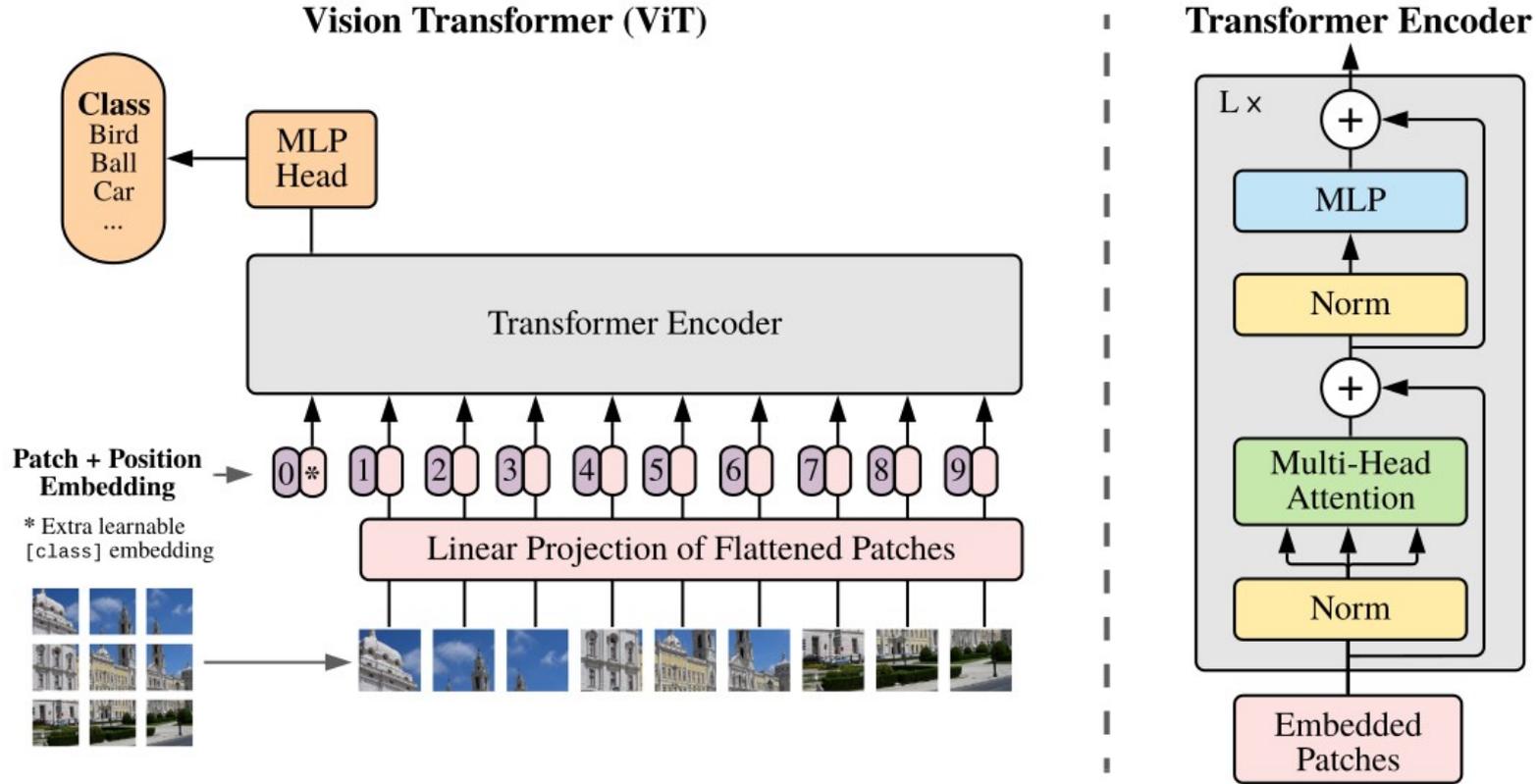
# Encodage de la position



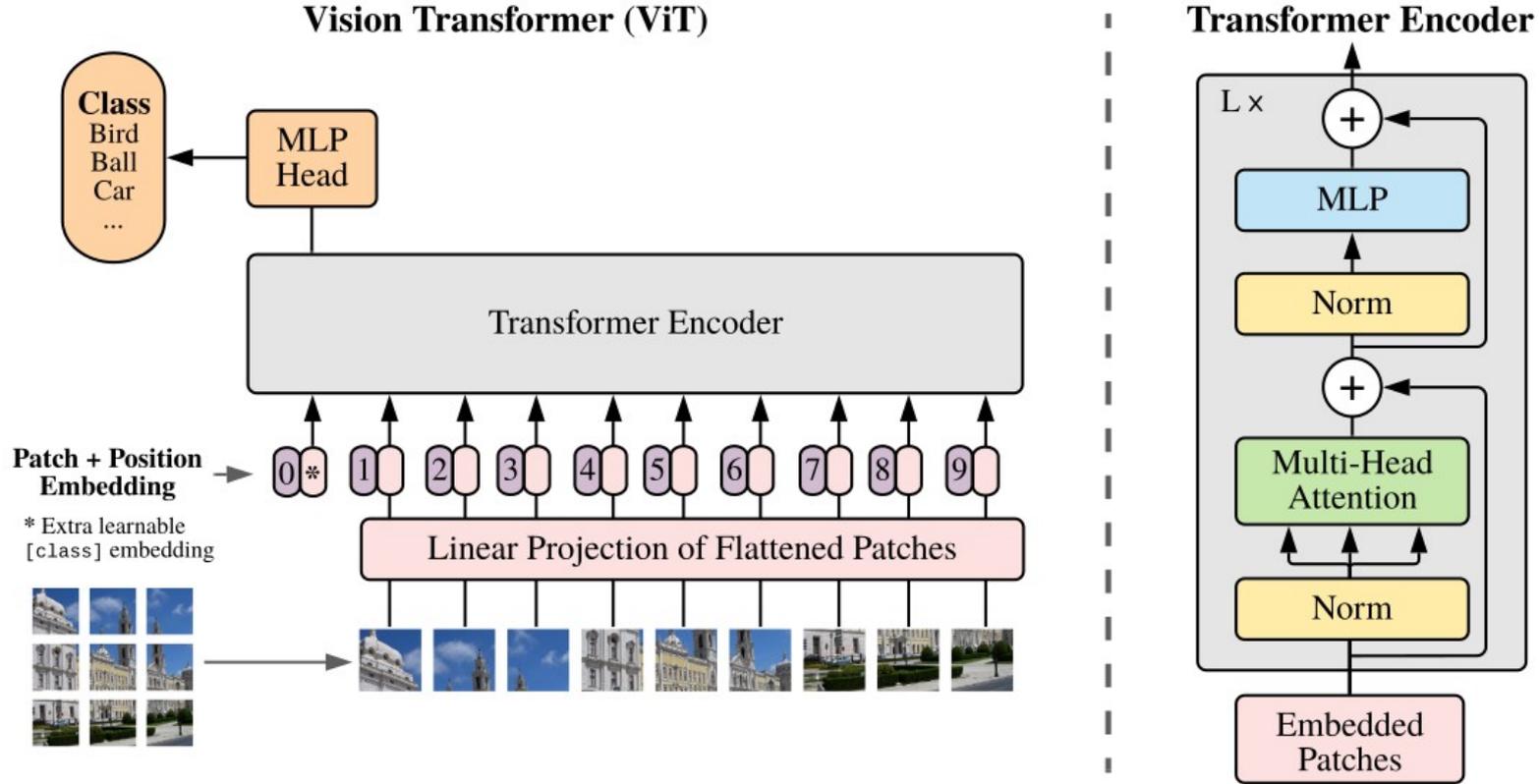
Remarque : Il existe d'autres façons d'encoder la position (ex: "Fourier Features").

## IV) Application à des images

# “An image is worth 16x16 words”, ICLR 2021



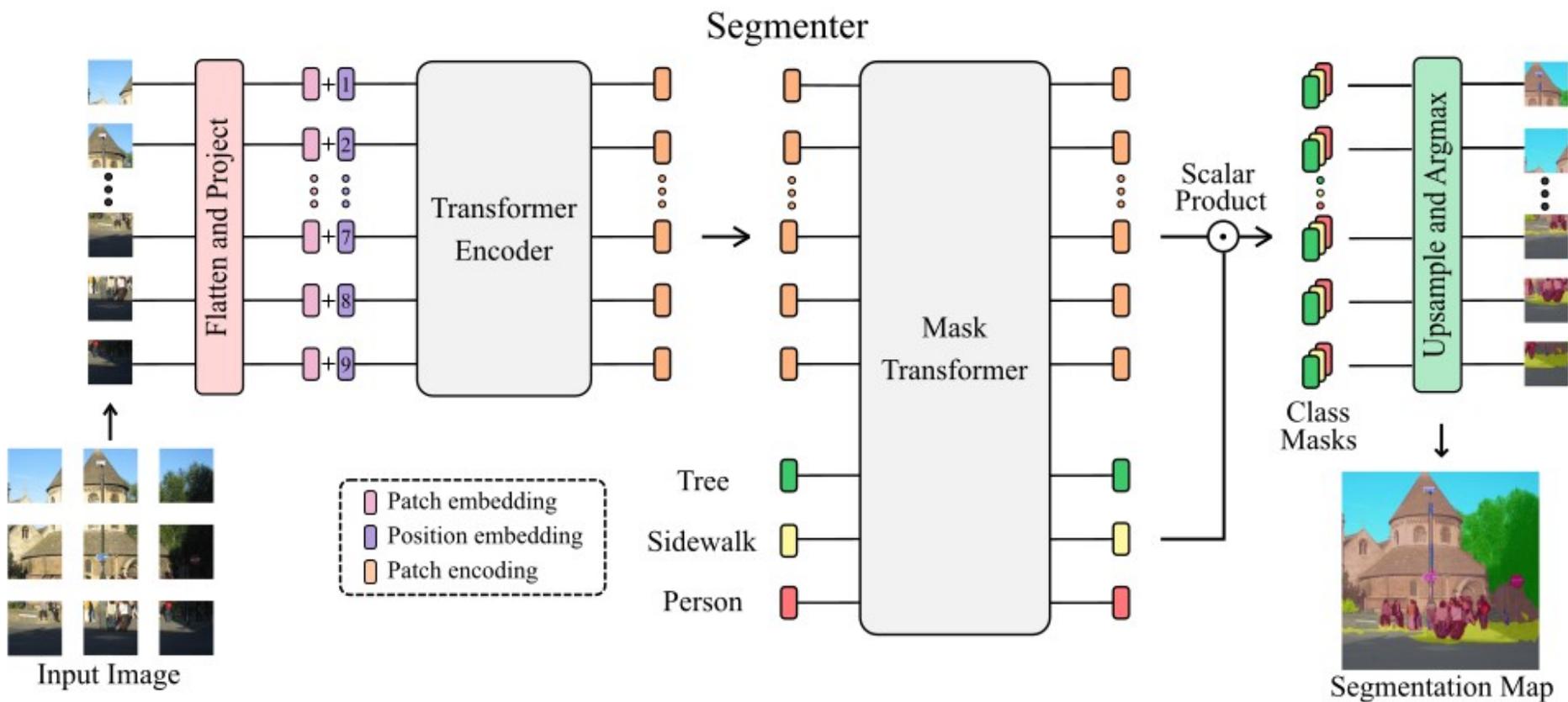
# “An image is worth 16x16 words”, ICLR 2021



Architecture ViT très utilisée de nos jours.

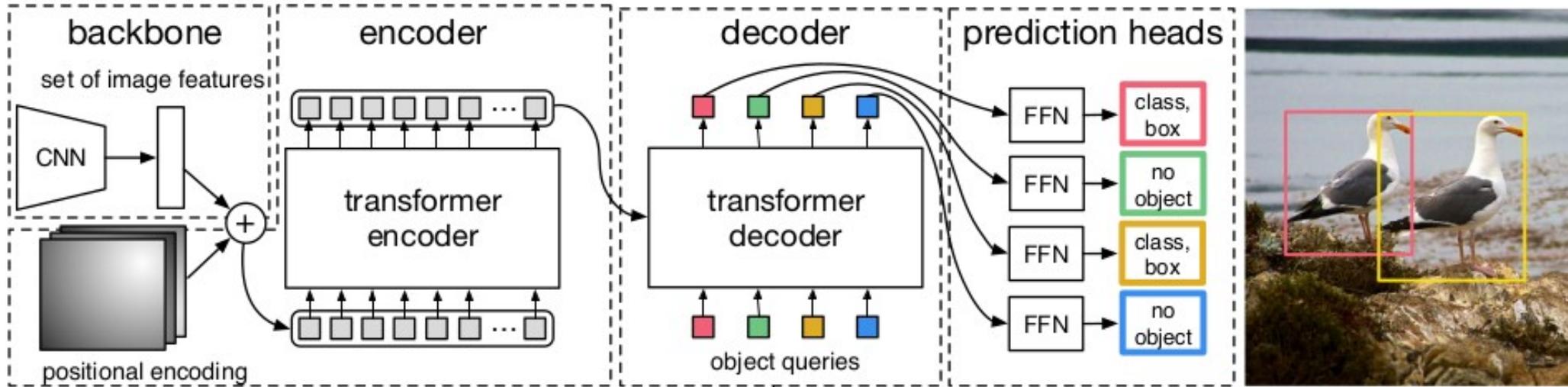
IV)

# “Segmenter: Transformer for Semantic Segmentation”, ICCV 2021



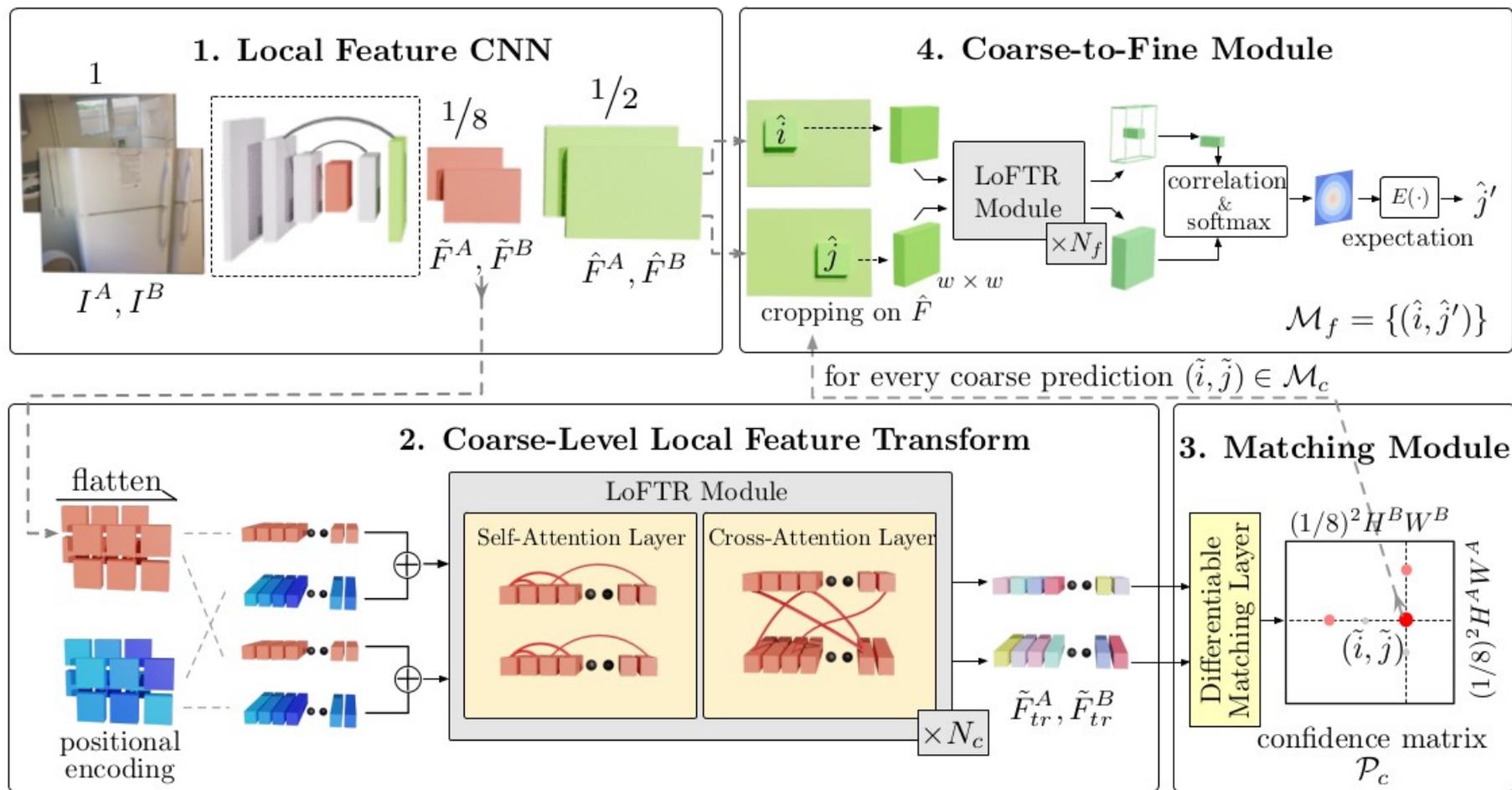
IV)

# “DETR: End-to-End Object Detection With Transformers”, ECCV 2020



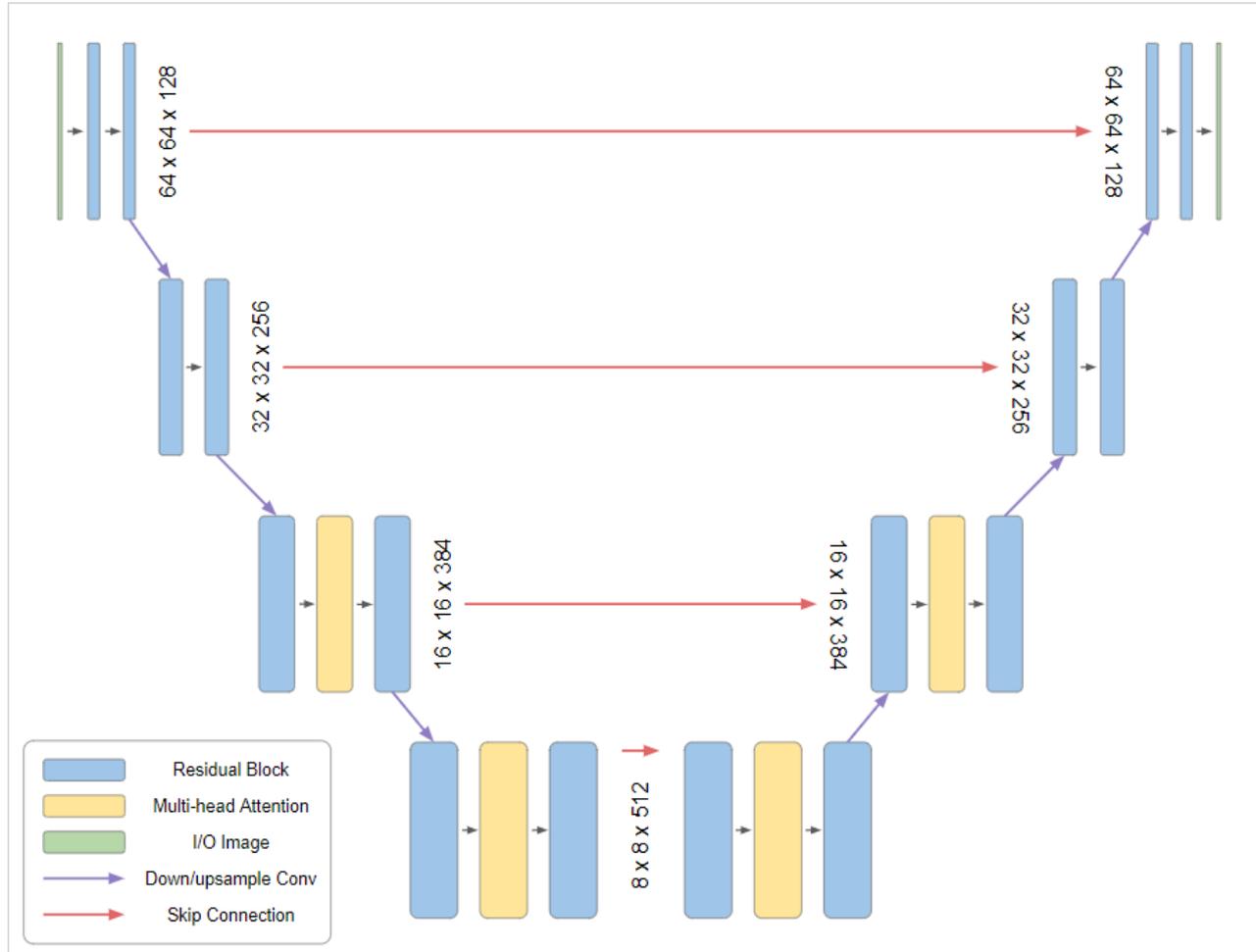
En pratique, il y a 100 “object queries”, donc 100 boîtes englobantes prédites.

# “LoFTR: Detector-Free Local Feature Matching with Transformers”, CVPR 2021



IV)

# U-Net avec des blocs d'attention



Architecture très utilisée par les méthodes de diffusion (prochain cours)

# V) Limites

v)

## Limites des couches d'attention à softmax

$$M = XQ(YK)^{\top} : N_x \times N_y \quad S = \text{softmax}(M, \text{dim}=1) : N_x \times N_y$$

Problème : Inapplicable pour des ensembles de grandes tailles.

v)

## Limites des couches d'attention à softmax

$$M = XQ(YK)^T : N_x \times N_y \quad S = \text{softmax}(M, \text{dim}=1) : N_x \times N_y$$

Problème : Inapplicable pour des ensembles de grandes tailles.

Solutions :

- Appliquer des couches d'attention à softmax en cherchant à réduire  $N_x$  et/ou  $N_y$   
→ **ViT** ou **U-Net avec attention**
- Modifier la couche d'attention softmax  
→ peu utilisé en pratique

v)

# Exemple de modification de la couche d'attention softmax : couche d'attention linéaire

“Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”, ICML 2020

$$\mathbf{x}' = \mathbf{x} + \left( \sum_{j=1}^{N_y} \frac{\phi(\mathbf{x}\mathbf{Q})\phi(\mathbf{y}_j\mathbf{K})^\top}{\sum_{k=1}^{N_y} \phi(\mathbf{x}\mathbf{Q})\phi(\mathbf{y}_k\mathbf{K})^\top} \mathbf{y}_j \right) \mathbf{V}$$

Remplacement du noyau exponentiel par un noyau linéaire

Se simplifie en

$$\mathbf{x}' = \mathbf{x} + \phi(\mathbf{x}\mathbf{Q}) \frac{\sum_{j=1}^{N_y} \phi(\mathbf{y}_j\mathbf{K})^\top \mathbf{y}_j \mathbf{V}}{\phi(\mathbf{x}\mathbf{Q}) \sum_{k=1}^{N_y} \phi(\mathbf{y}_k\mathbf{K})^\top}$$

Indépendant de  $\mathbf{x}$ , plus besoin de calculer ni stocker explicitement les matrices  $\mathbf{M}$  et  $\mathbf{S}$

v)

# Exemple de réduction de $N_x$ et/ou $N_y$ : PerceiverIO

“Perceiver IO: A General Architecture for Structured Inputs & Outputs.” arXiv, 2021

