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 et tendances actuelles

I) Histoire du “Transformer”

“Attention is All You Need”, NIPS 2017

Sequence-to-sequence

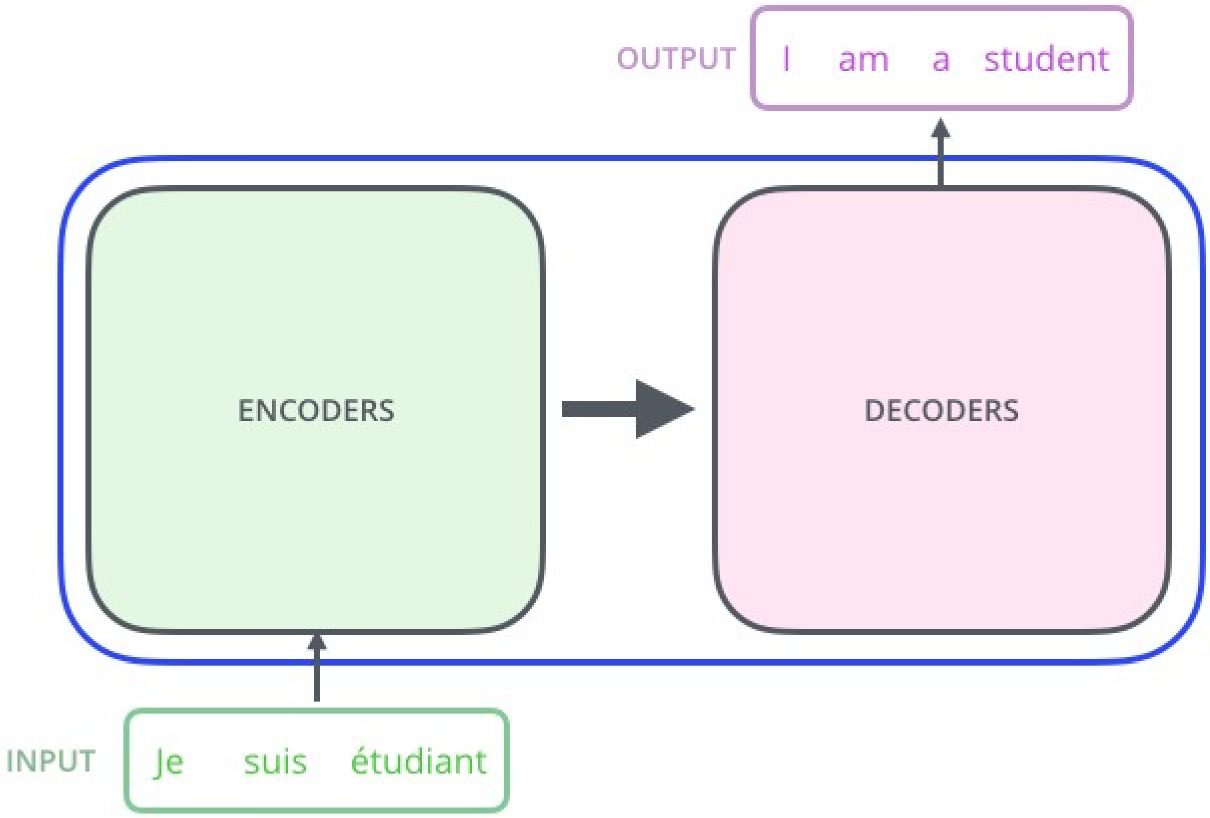
[x1, x2, x3, …, xn] → Seq2Seq → [y1, y2, y3, …, ym]

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

I)

Architecture encodeur-décodeur

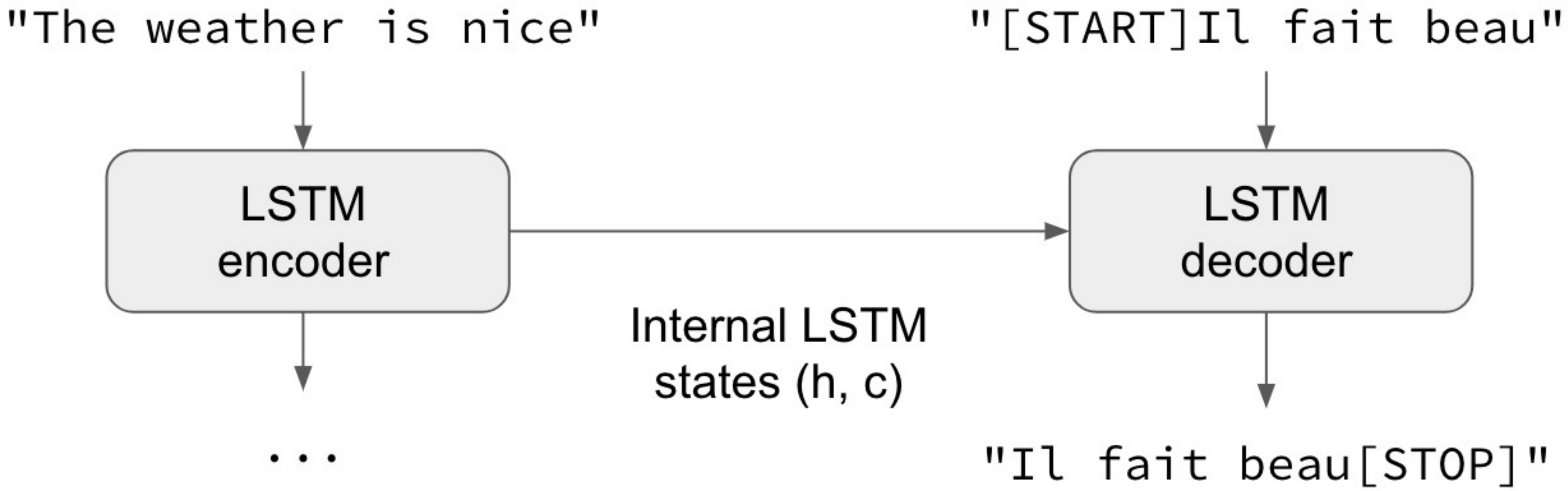
Encodeur :
- RNN
- ou CNN
- ou Transformer



Décodeur :
- RNN
- ou CNN
- ou Transformer

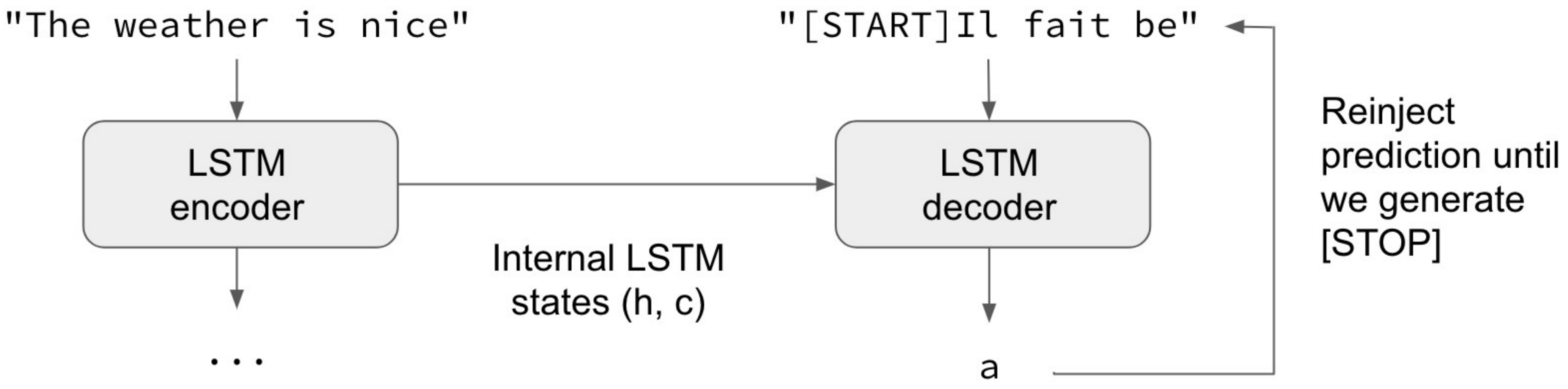
Exemple avec un RNN

Apprentissage (en “Teacher-Forcing”)



Exemple avec un RNN (suite)

Inférence



I)

Exemple avec un RNN (suite)

Limites

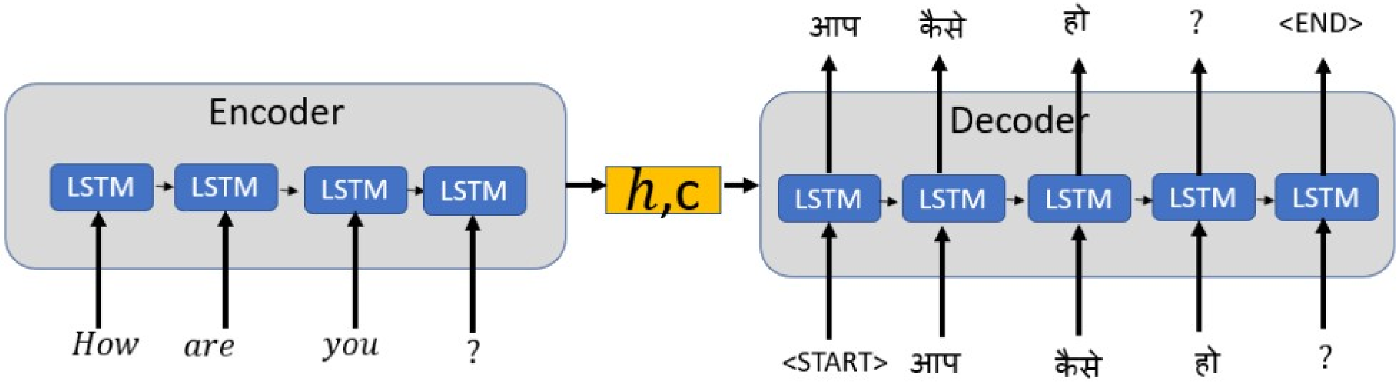
Calcul séquentiel

↓

Apprentissage lent

↓

Comment paralléliser ?

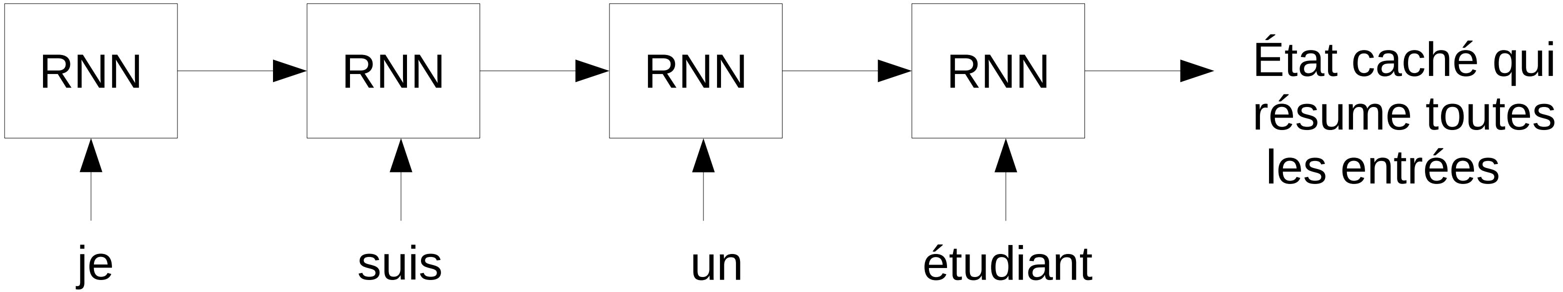


Difficile d'apprendre de longues dépendances avec un RNN.

Transformer vs RNN

Encodeur RNN

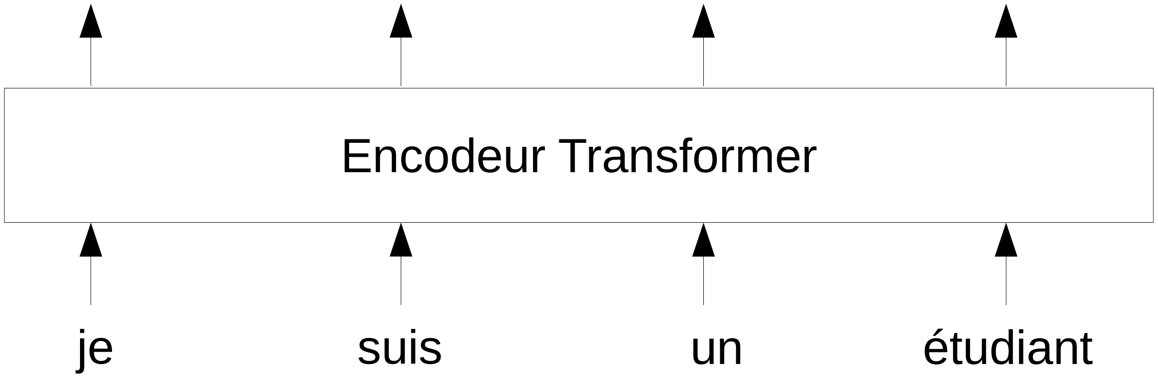
Calcul séquentiel
+
Chemin de la longueur de la phrase d'entrée



Encodeur Transformer

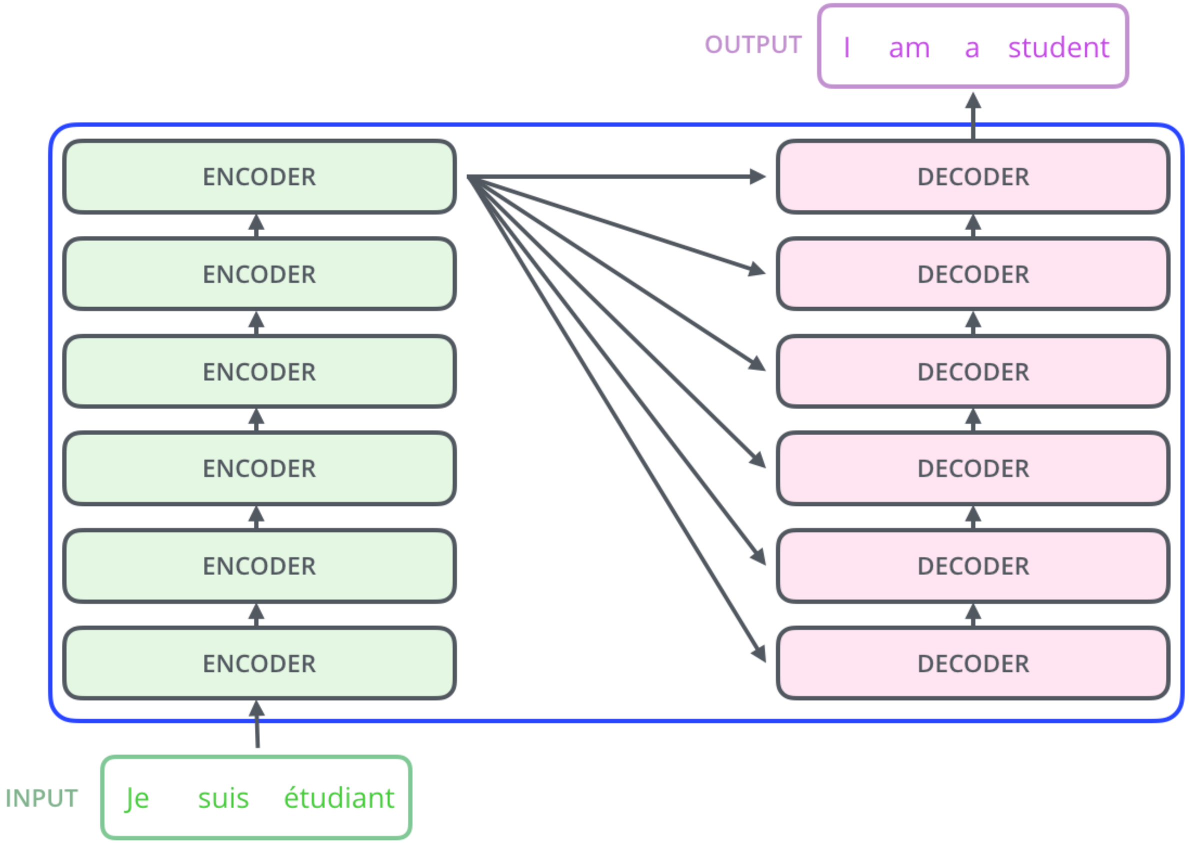
Calcul en parallèle
+
Chemin de longueur constante

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



I)

Transformer : vue globale



I)

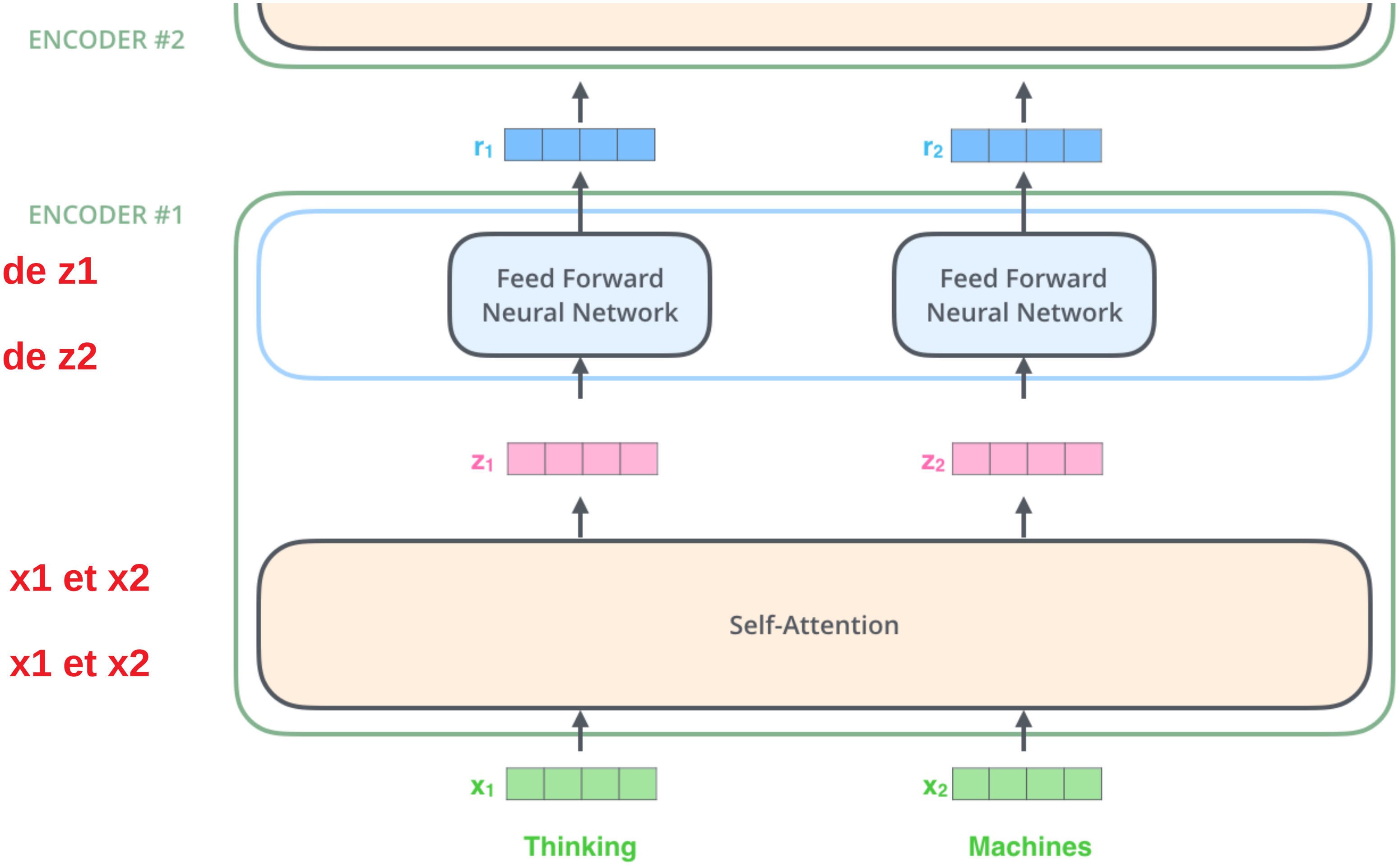
Transformer : encodeur

r1 dépend de z1

r2 dépend de z2

z1 dépend de x1 et x2

z2 dépend de x1 et x2



II) Couche d’attention à softmax

II)

Attention utilisant la fonction softmax

Entrées	$\mathbf{x}$	: vecteur de dimension	$1 \times D_x$
	$\{\mathbf{y}_i\}_{i=1\ldots N_y}$	: ensemble de vecteurs de dimension	$1 \times D_y$
Sortie	$\mathbf{x}'$	: vecteur de dimension	$1 \times D_x$
Paramètres	$\mathtt{Q}$	: matrice “query” de taille	$D_x \times L$
	$\mathtt{K}$	: matrice “key” de taille	$D_y \times L$
	$\mathtt{V}$	: matrice “value” de taille	$D_y \times D_x$

Fonction

$$\mathbf{x}' = \mathbf{x} + \sum_{j=1}^{N_y} \underbrace{\boxed{\frac{\exp(\mathbf{x}\mathtt{Q}(\mathbf{y}_j\mathtt{K})^\top)}{\sum_{k=1}^{N_y}\exp(\mathbf{x}\mathtt{Q}(\mathbf{y}_k\mathtt{K})^\top)}}}_{\textbf{Softmax}} \mathbf{y}_j\mathtt{V}$$

scalaire

Attention utilisant la fonction softmax (suite)

$$\mathbf{x}' = \mathbf{x} + \left(\sum_{j=1}^{N_y} s_j \mathbf{y}_j \right) \mathtt{V} \qquad \text{où} \qquad s_j = \frac{\exp(\mathbf{x}\mathtt{Q}(\mathbf{y}_j\mathtt{K})^\top)}{\sum_{k=1}^{N_y} \exp(\mathbf{x}\mathtt{Q}(\mathbf{y}_k\mathtt{K})^\top)}$$

Transfert d'information depuis $\{\mathbf{y}_i\}_{i=1\ldots N_y}$ vers $\mathbf{x}$ le tout stocké dans $\mathbf{x}'$

$\mathtt{Q}$
$\mathtt{K}$ permettent d'apprendre à transférer l'information pertinente pour la tâche concernée
$\mathtt{V}$

Terminologie : "Dot-product attention", plus rarement "softmax attention"

II)

Inter-attention ("Cross-attention")

Entrées $\{\mathbf{x}_i\}_{i=1\ldots N_x}$: vecteurs de dimension D_x $\longrightarrow$ $\mathtt{X}$: matrice de taille $N_x \times D_x$

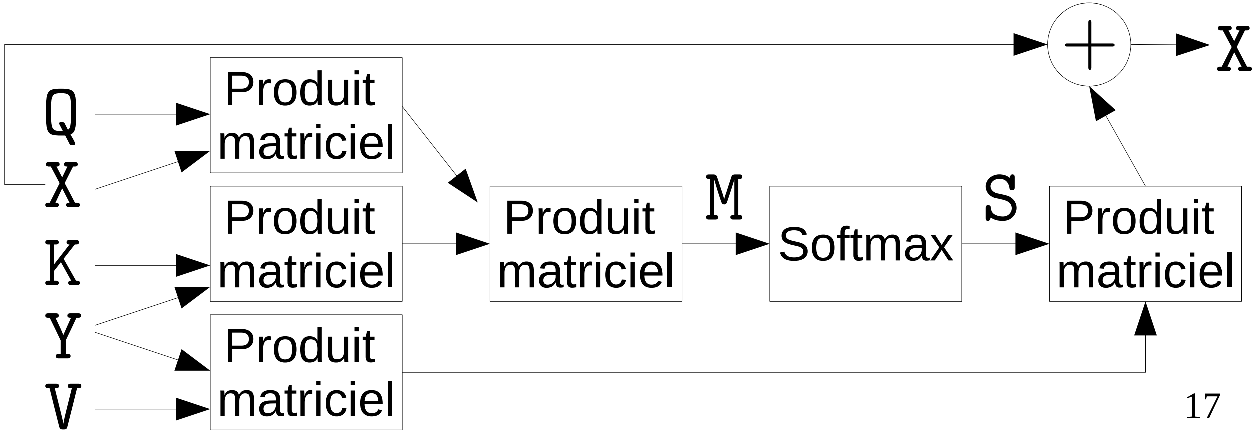
$\{\mathbf{y}_i\}_{i=1\ldots N_y}$: vecteurs de dimension D_y $\longrightarrow$ $\mathtt{Y}$: matrice de taille $N_y \times D_y$

Sorties $\{\mathbf{x}'_i\}_{i=1\ldots N_x}$: vecteurs de dimension D_x $\longrightarrow$ $\mathtt{X}'$: matrice de taille $N_x \times D_x$

$$\mathtt{M} = \mathtt{XQ}(\mathtt{YK})^\top$$

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

$$\mathtt{X}' = \mathtt{X} + \mathtt{SYV}$$



Transfert d'information depuis $\mathtt{Y}$ vers $\mathtt{X}$ le tout stocké dans $\mathtt{X}'$

II)

Inter-attention (“Cross-attention”) (suite)

$$\mathtt{X} : N_x \times D_x \quad \mathtt{Y} : N_y \times D_y \quad \mathtt{X}' : N_x \times D_x$$

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

Calcul

- **Nombre d’opérations potentiellement très élevé**

+ **Parallélisable**

Stockage

- **Mémoire requise pour stocker M et S potentiellement très élevée**

Cas particulier : Auto-attention ("Self-attention")

Entrées $\{\mathbf{x}_i\}_{i=1\ldots N_x}$: vecteurs de dimension D_x $\longrightarrow$ $\mathtt{X}$: matrice de taille $N_x \times D_x$

Sorties $\{\mathbf{x}'_i\}_{i=1\ldots N_x}$: vecteurs de dimension D_x $\longrightarrow$ $\mathtt{X}'$: matrice de taille $N_x \times D_x$

$$\mathtt{M} = \mathtt{XQ}(\mathtt{XK})^\top \ : \ N_x \times N_x$$
$$\mathtt{S} = \mathrm{softmax}(\mathtt{M}, \mathrm{dim}=1) \ : \ N_x \times N_x$$
$$\mathtt{X}' = \mathtt{X} + \mathtt{SXV} \ : \ N_x \times D_x$$

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

II)

Couche d'attention softmax à têtes multiples ”Multi-head dot-product attention”

$$\mathtt{X} : N_x \times D_x \quad \mathtt{Y} : N_y \times D_y \quad \mathtt{X}' : N_x \times D_x$$

Pour $k = 1$ à H

1) H couches d'attention indépendantes (têtes)

$$\mathtt{M}_k = \mathtt{X}\mathtt{Q}_k(\mathtt{Y}\mathtt{K}_k)^\top : N_x \times N_y$$

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

$$\mathtt{Z}_k = \mathtt{S}_k\mathtt{Y}\mathtt{V}_k : N_x \times D_v$$

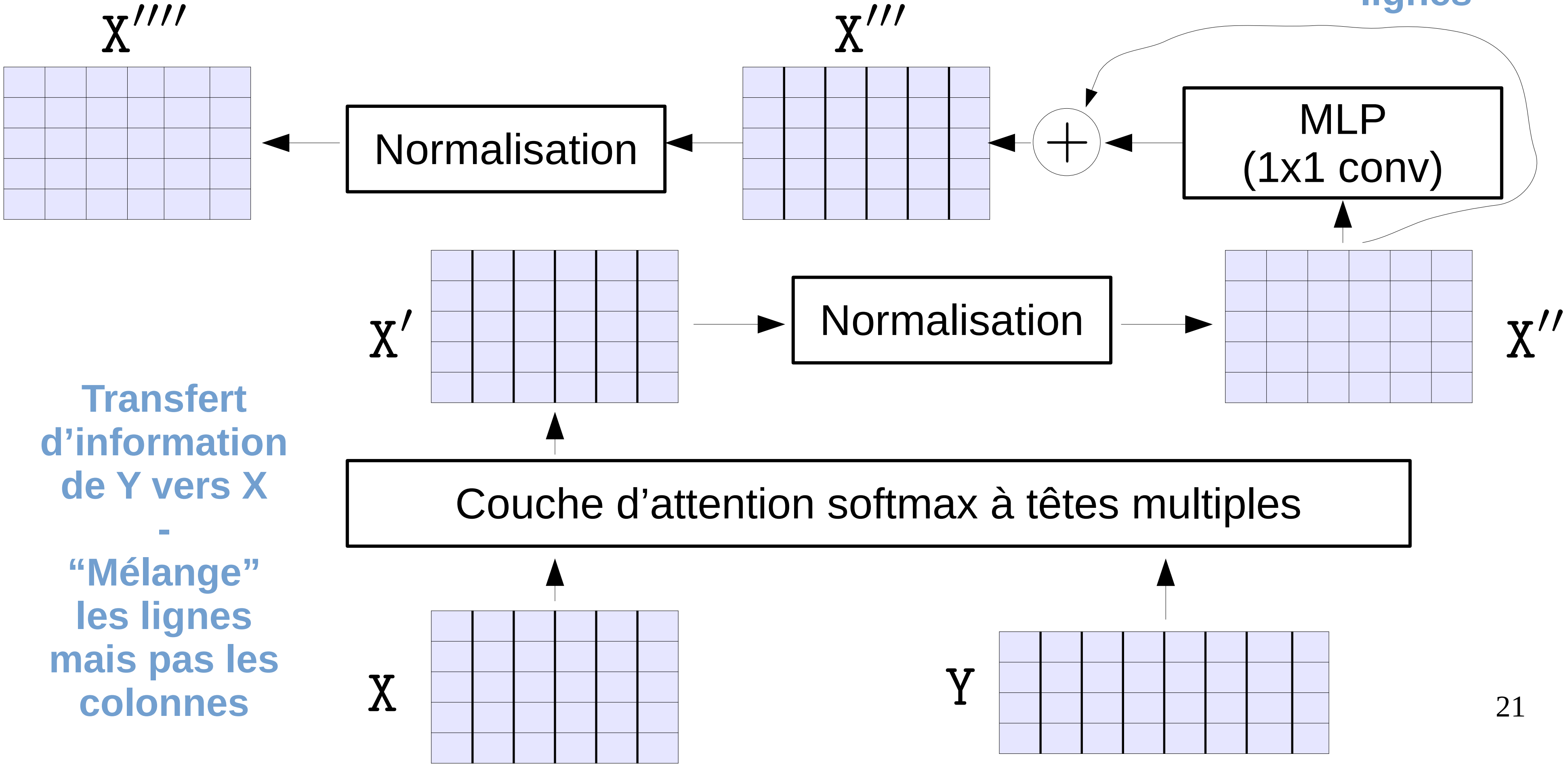
$$\mathtt{Z} = [\mathtt{Z}_1\,\mathtt{Z}_1 \ldots \mathtt{Z}_H] : N_x \times (H \times D_v)$$

2) “mélange” des résultats des H têtes

$$\mathtt{X}' = \mathtt{X} + \mathtt{Z}\mathtt{W} : N_x \times D_x$$

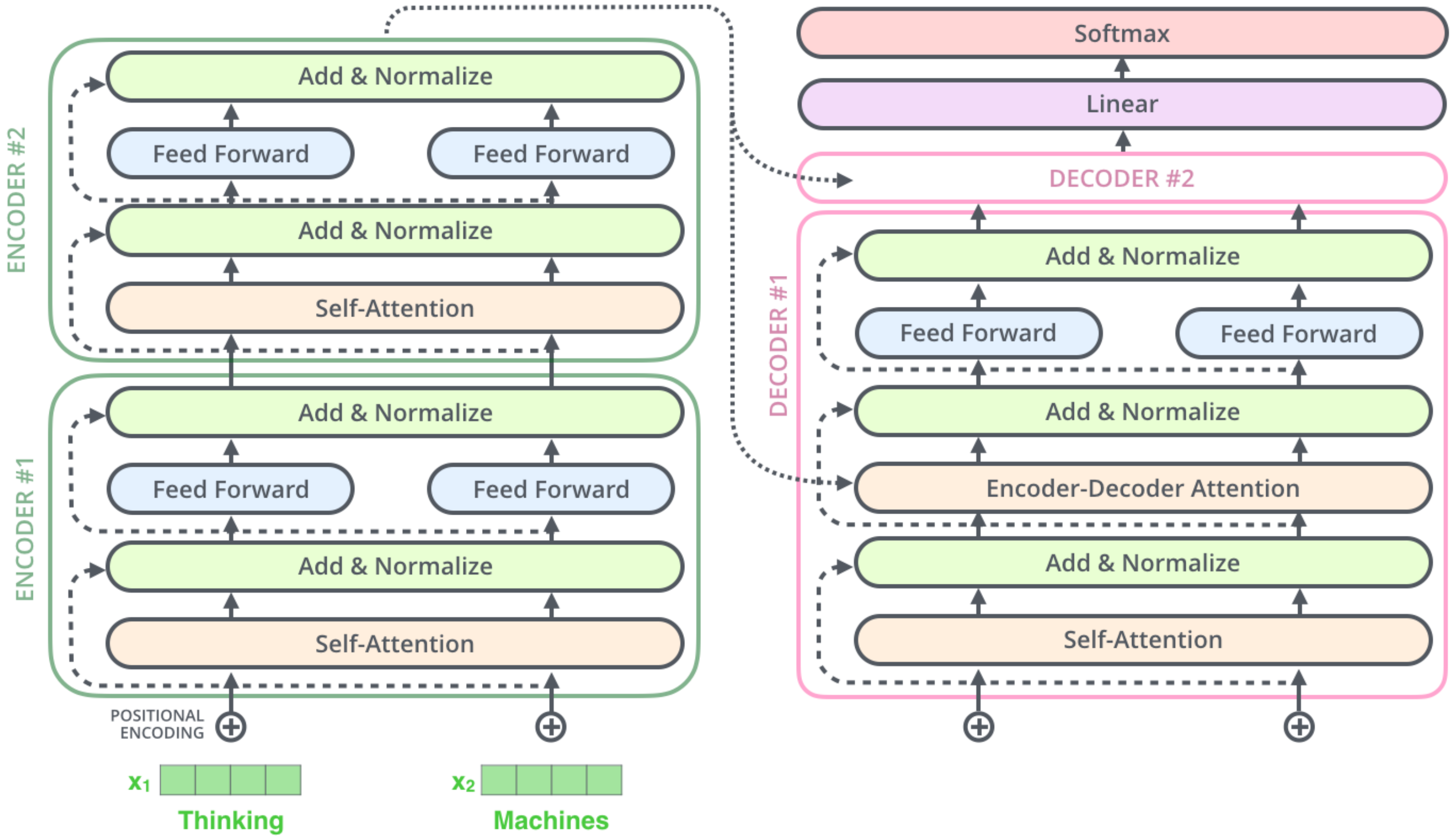
II)

Bloc d’attention “classique”



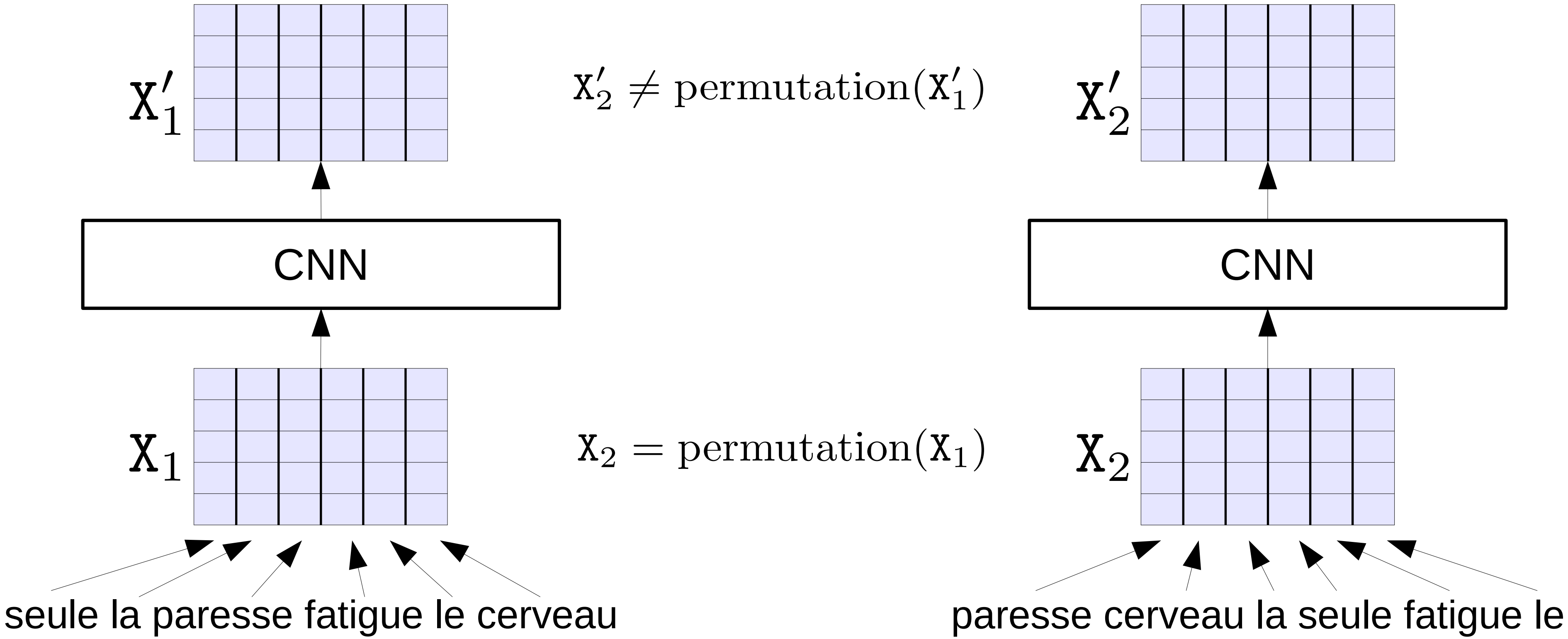
II)

Vue détaillée du Transformer



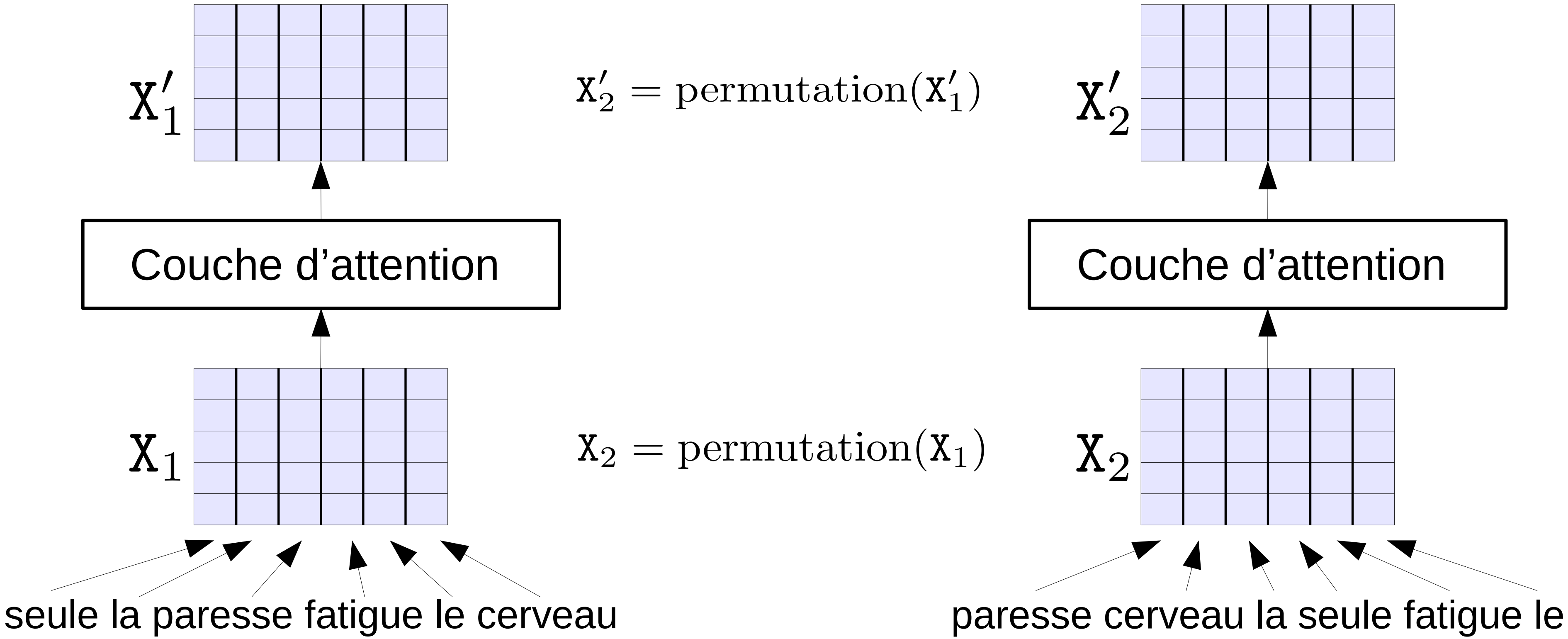
III) Équivariance par permutation et encodage de la position

Équivariance par permutation



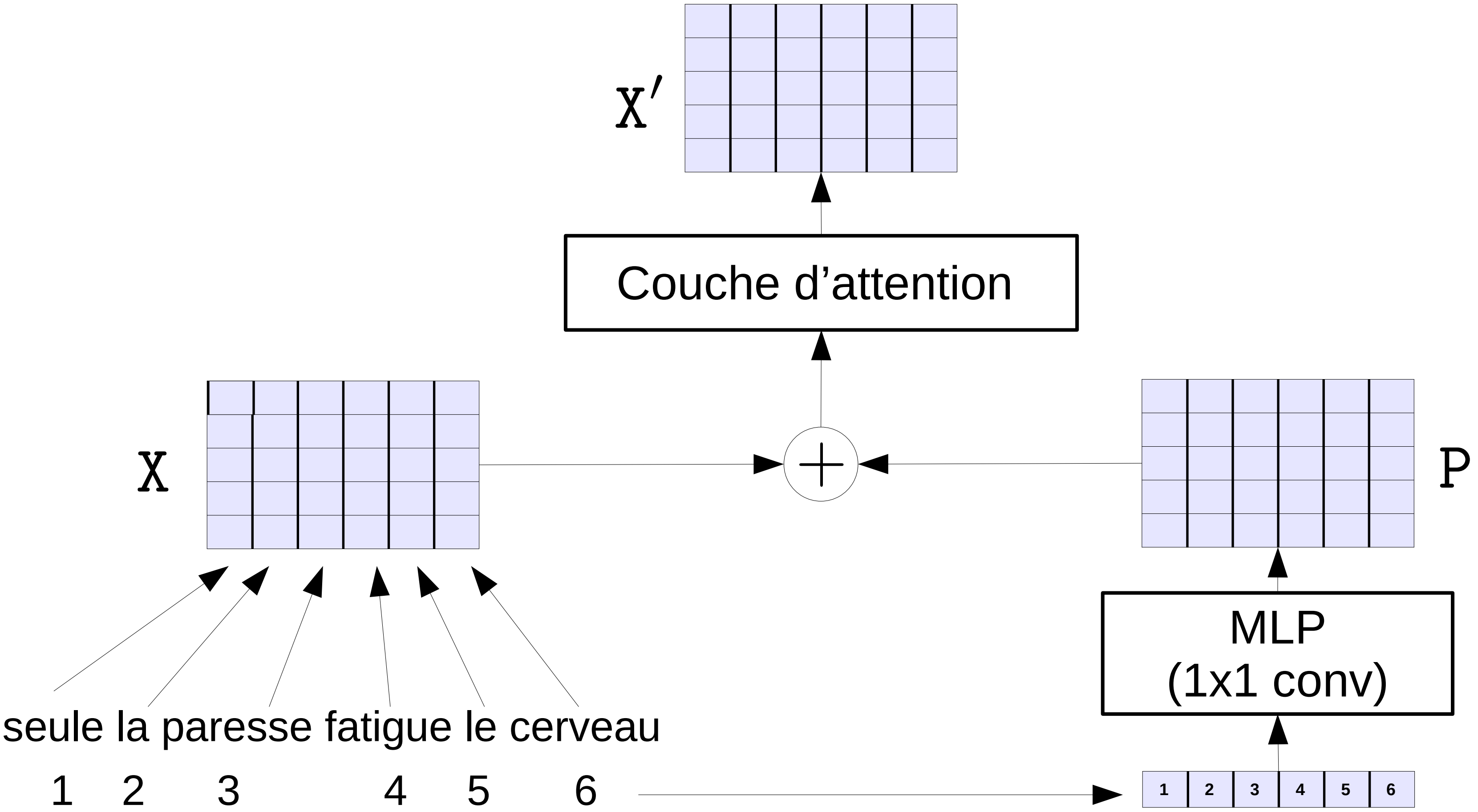
CNN adapté aux ensembles ordonnés (phrase, signal, image, etc.)
→ ses filtres s'appliquent sur un voisinage

Équivariance par permutation (suite)



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

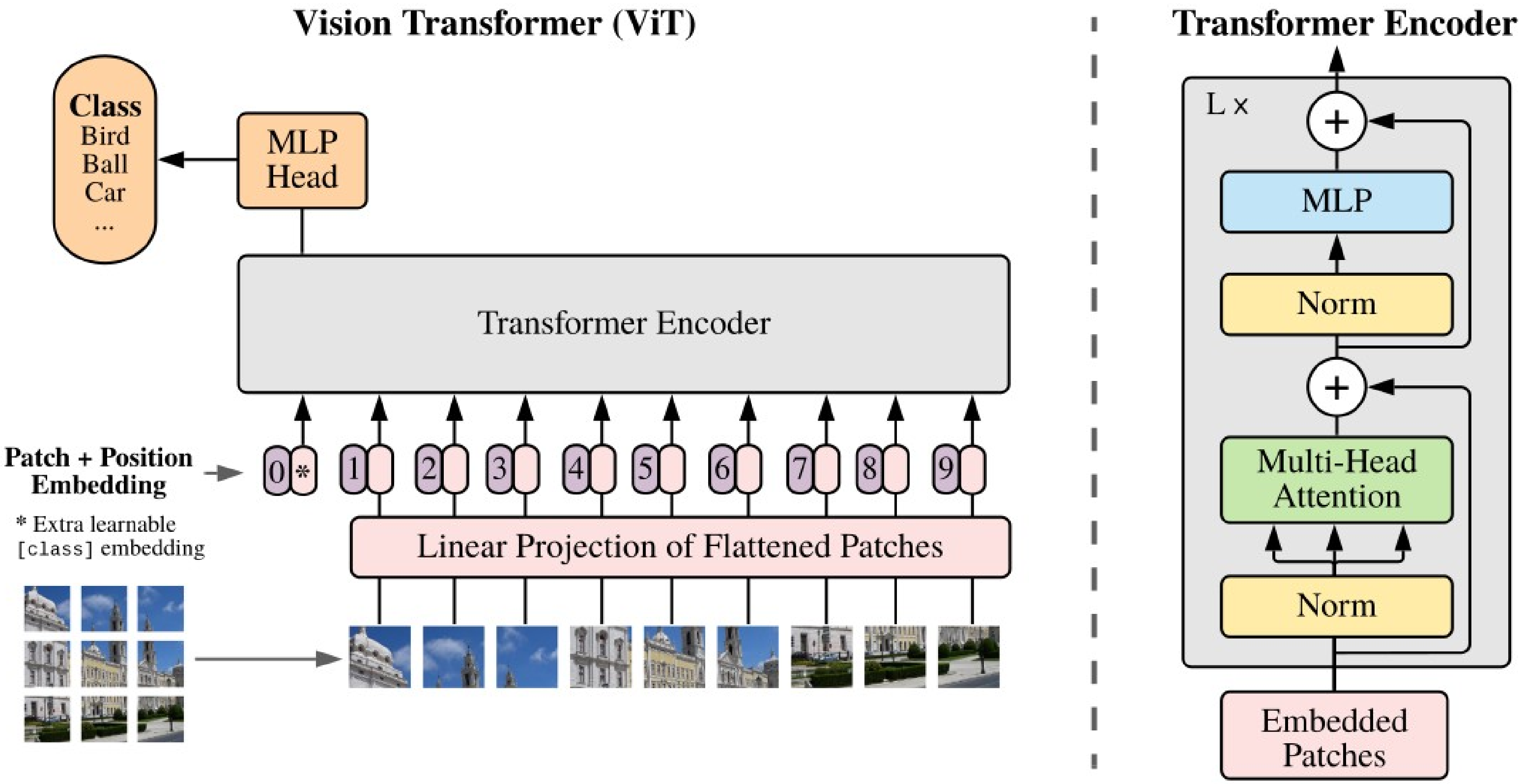
Encodage de la position



IV) Application à des images

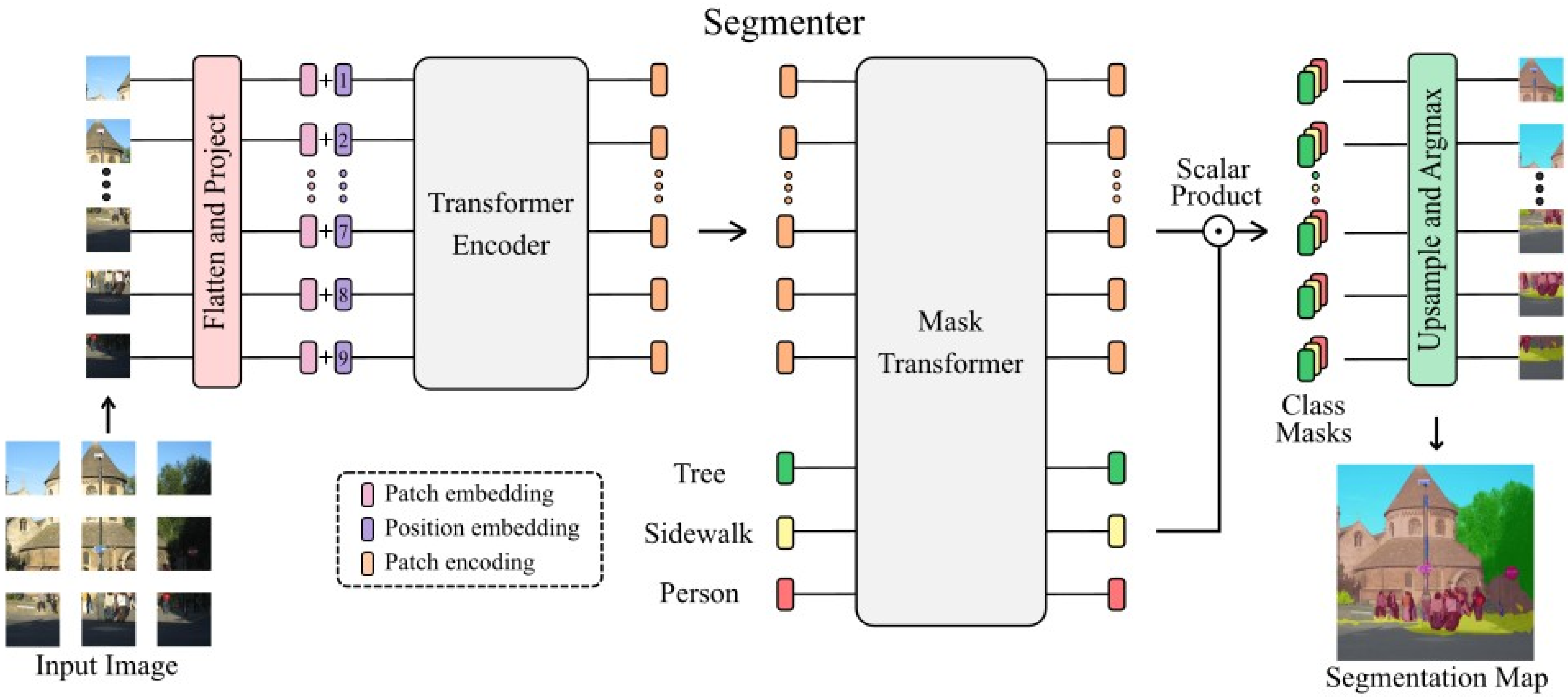
IV)

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



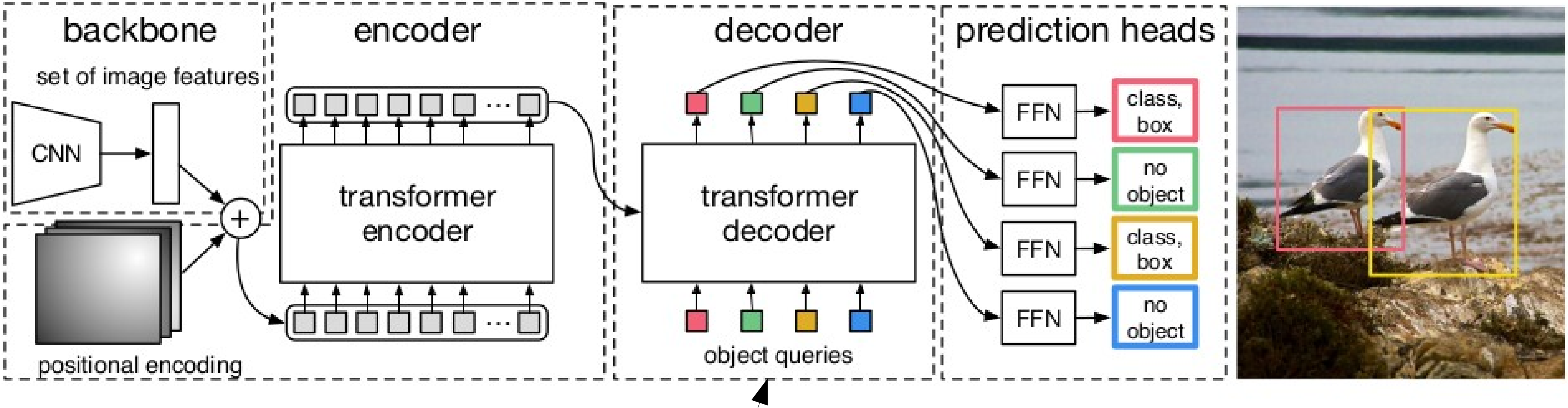
IV)

"Segmenter: Transformer for Semantic Segmentation", ICCV 2021



IV)

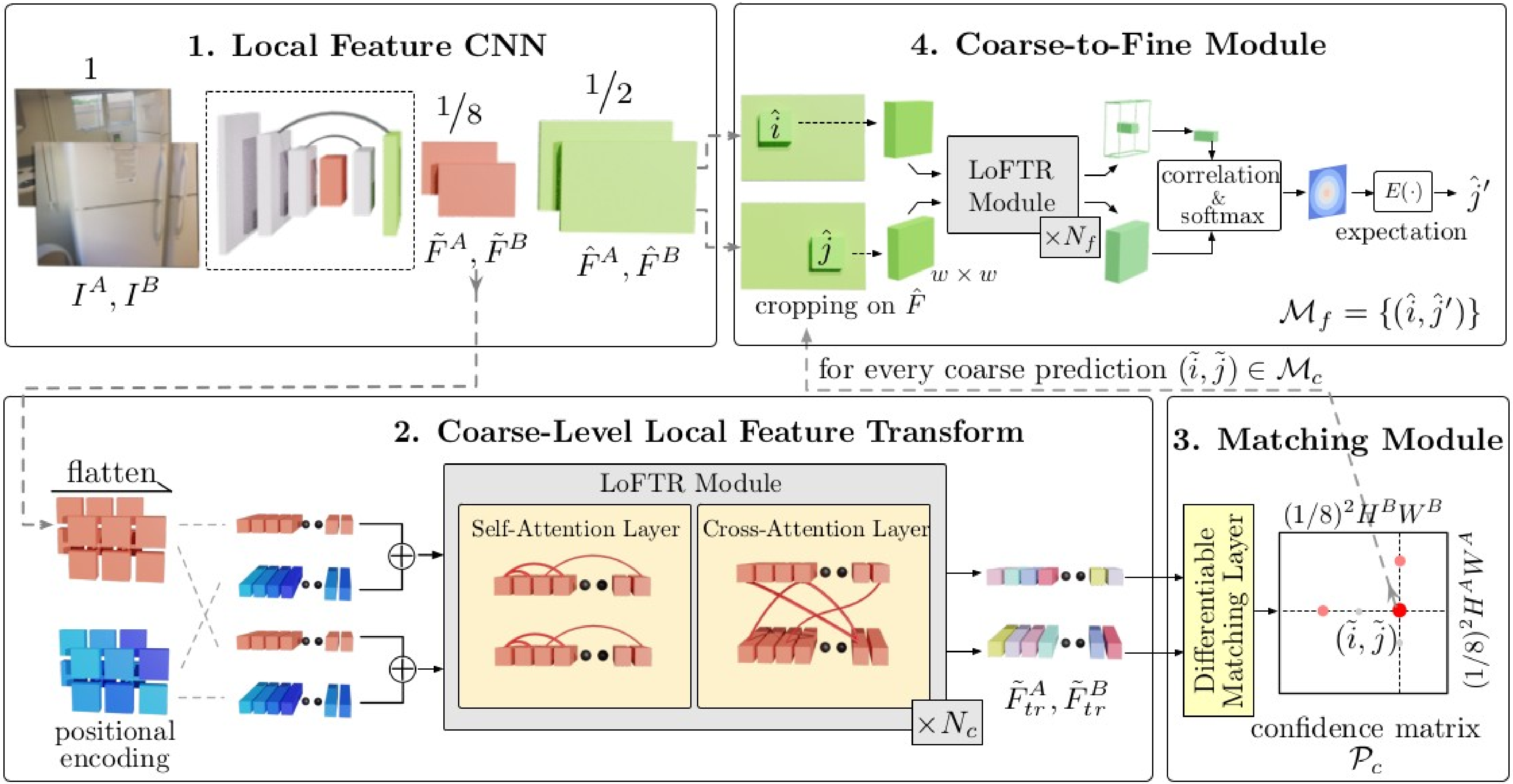
“DETR: End-to-End Object Detection With Transformers”, ECCV 2020



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

IV)

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



V) Limites et tendances actuelles

Limites des couches d'attention à softmax

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

Problème : Inapplicable pour des ensembles de grandes tailles.

Solutions :

- Appliquer des couches d'attention à softmax en cherchant à réduire Nx et/ou Ny
- Modifier la couche d'attention softmax

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

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

Remplacement du noyau exponentiel par un noyau linéaire

Se simplifie en

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

Indépendant de x, plus besoin de calculer ni stocker explicitement les matrices M et S

V)

Exemple de réduction de Nx et/ou Ny : PerceiverIO

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