Axe transversale Intelligence Artificielle

Tutorial

Mécanisme d'attention et Transformers

Blaise Hanczar

Laboratoire IBISC
Informatique, Bioinformatique et Systèmes Complexes

Etat de l'art (paperswithcode.com)

Classification d'image (ImageNet)

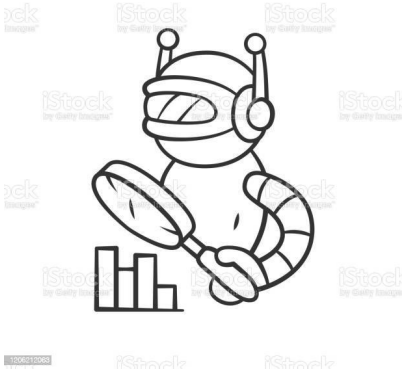
Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**Model soups** (ViT-G/14)	90.94%		1843M	✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time			2022	Transformer, JFT-3B
2	**CoAtNet-7**	90.88%		2440M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
3	**ViT-G/14**	90.45%		1843M	✓	Scaling Vision Transformers			2021	Transformer, JFT-3B
4	**CoAtNet-6**	90.45%		1470M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
5	**DaViT-G**	90.4%		1437M	✓	DaViT: Dual Attention Vision Transformers			2022	Transformer
6	**Meta Pseudo Labels** (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels			2021	EfficientNet, JFT-300M
7	**DaViT-H**	90.2%		362M	✓	DaViT: Dual Attention Vision Transformers			2022	Transformer
8	**SwinV2-G**	90.17%			✓	Swin Transformer V2: Scaling Up Capacity and Resolution			2021	Transformer
9	**Florence-CoSwin-H**	90.05%	99.02%		✓	Florence: A New Foundation Model for Computer Vision			2021	Transformer
10	**Meta Pseudo Labels** (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels			2021	EfficientNet, JFT-300M

Traduction Anglais-Français

Rank	Model	BLEU score	SacreBLEU	Extra Training Data	Paper
1	**Transformer+BT** (ADMIN init)	46.4	44.4	✓	Very Deep Transformers for Neural Machine Translation
2	**Noisy back-translation**	45.6	43.8	✓	Understanding Back-Translation at Scale
3	**mRASP+Fine-Tune**	44.3	41.7	✓	Pre-training Multilingual Neural Machine Translation by Leveraging Alignment Information
4	**Transformer + R-Drop**	43.95		×	R-Drop: Regularized Dropout for Neural Networks
5	**Transformer** (ADMIN init)	43.8	41.8	×	Very Deep Transformers for Neural Machine Translation
6	**Admin**	43.8		×	Understanding the Difficulty of Training Transformers
7	**BERT-fused NMT**	43.78		✓	Incorporating BERT into Neural Machine Translation
8	**MUSE** (Parallel Multi-scale Attention)	43.5		×	MUSE: Parallel Multi-Scale Attention for Sequence to Sequence Learning
9	**T5**	43.4		✓	Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
10	**Local Joint Self-attention**	43.3		×	Joint Source-Target Self Attention with Locality Constraints

Plan

- Attention
  - Modèle encodeur-décodeur
  - Mécanisme d’attention
  - Encodage de position



- Transformers
  - Architecture
  - Applications

Tutorial inspiré du cours de Fei-Fei Stanford cs231n

Image Captioning

- Génération automatique de légende d'image
- Entrée : image
- Sortie : texte

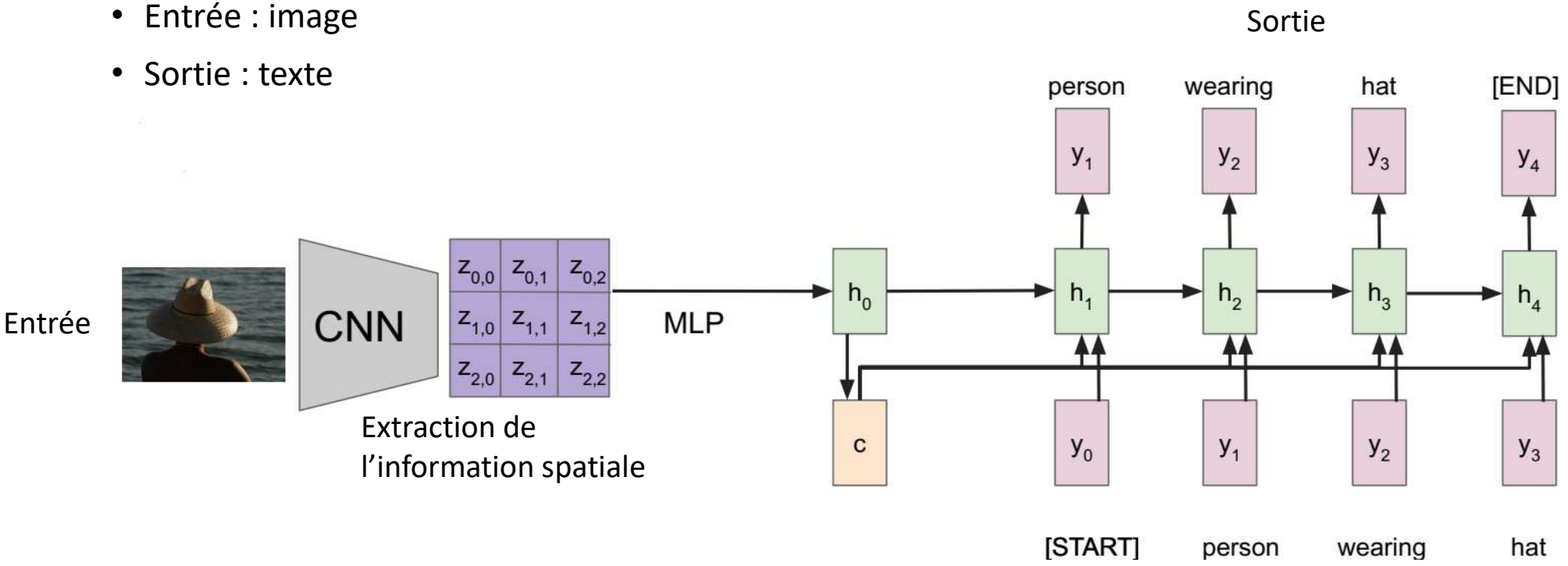


Image Captioning

- Encodeur : encode l'information de l'image dans un vecteur de contexte
- Décodeur : décode le vecteur de contexte en texte

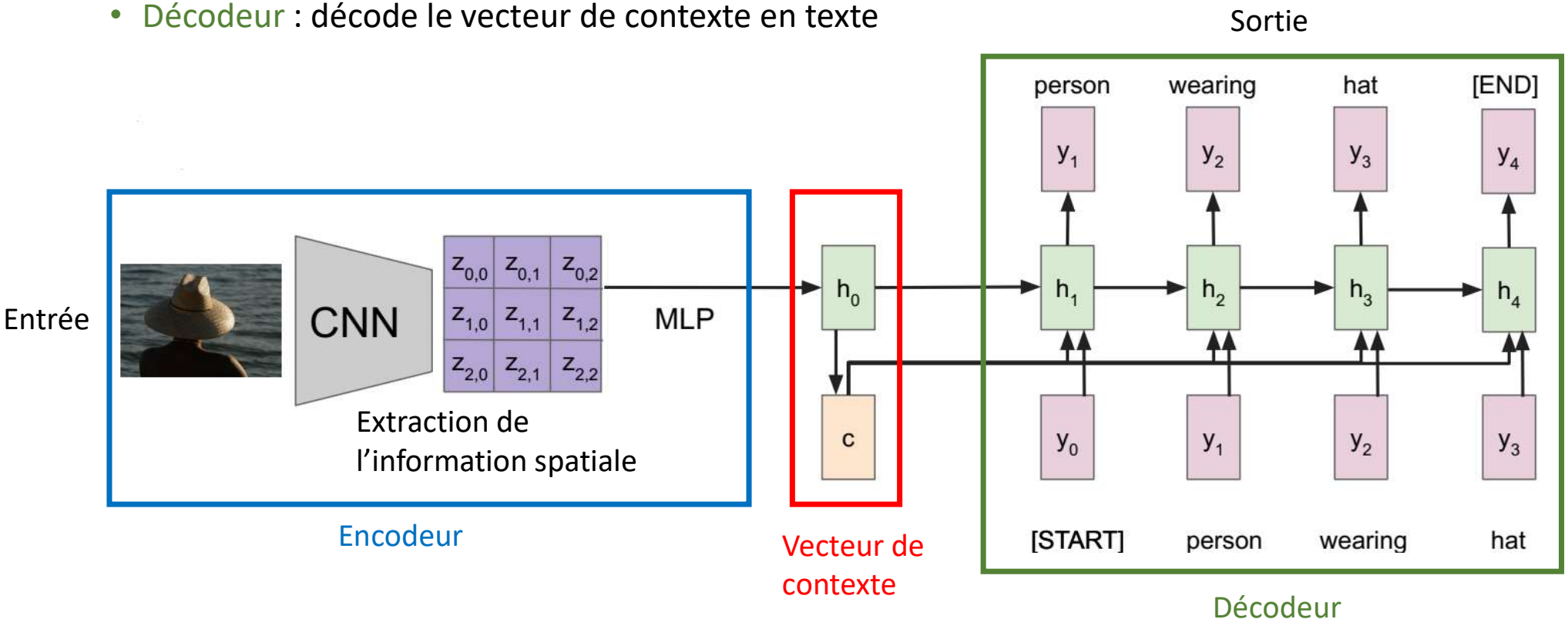


Image Captioning

- Encodeur : encode l'information de l'image dans un vecteur de contexte
- Décodeur : décode le vecteur de contexte en texte
- Goulot d'étranglement en c
  - Problème pour la génération de longue séquence

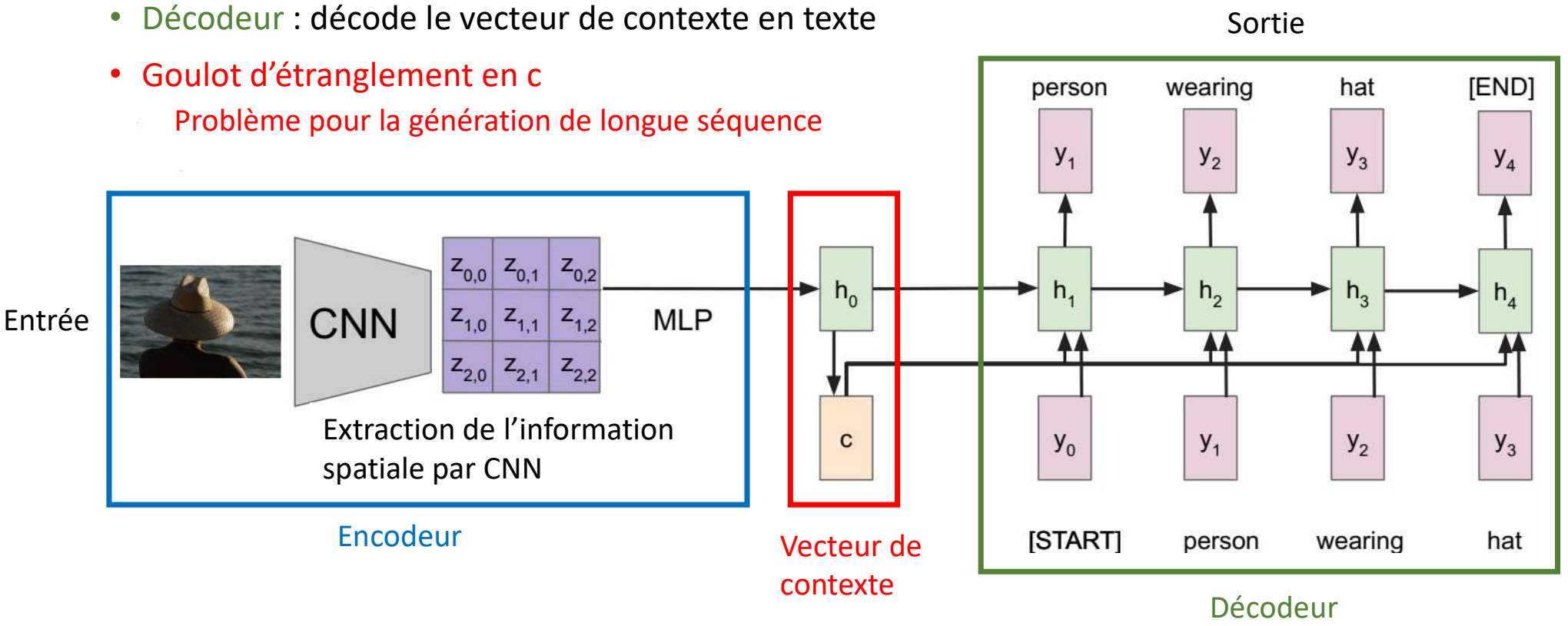
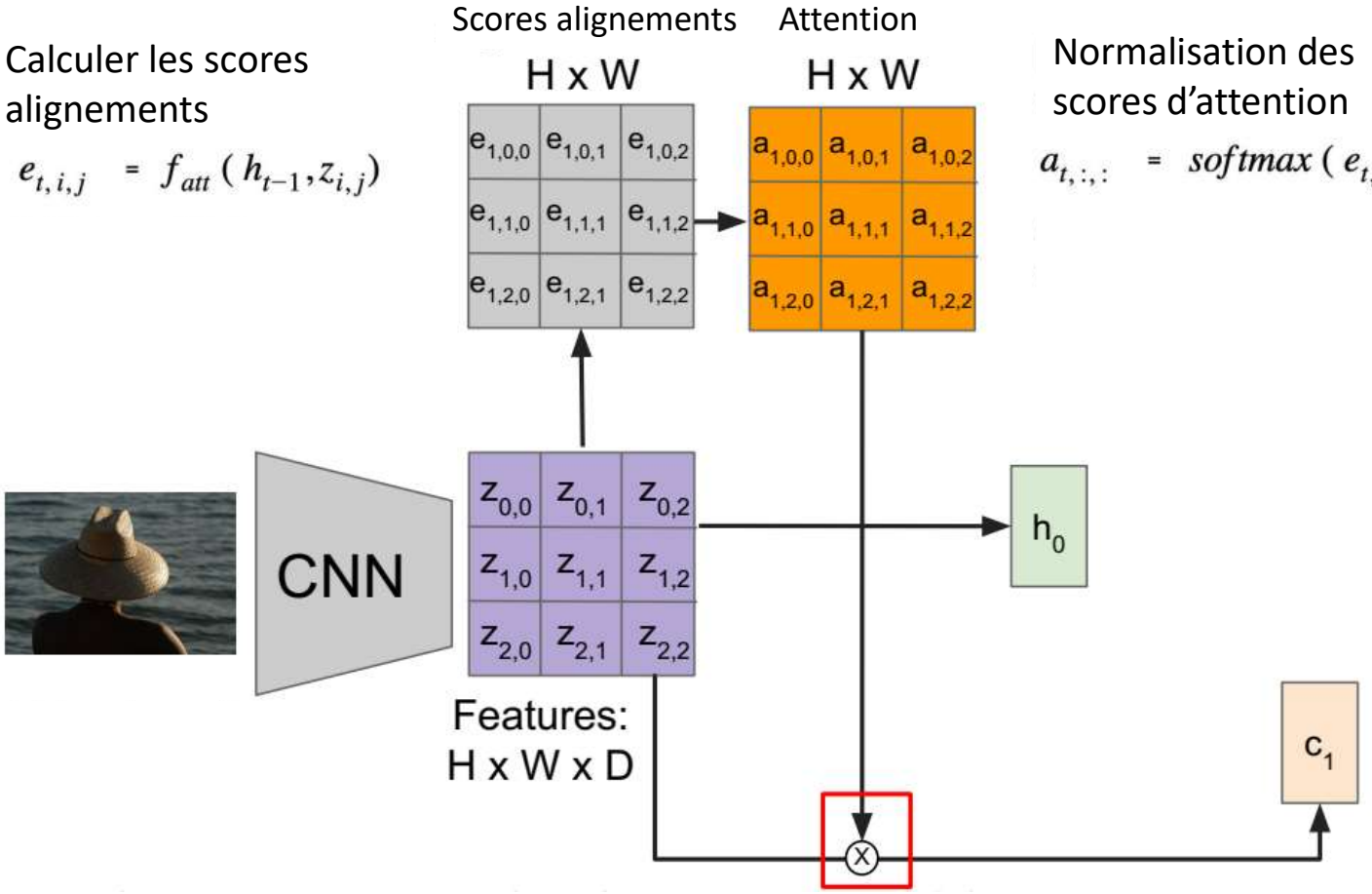


Image Captioning avec attention

Calculer les scores alignements

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$



Normalisation des scores d'attention

$$a_{t,:,:} = softmax(e_{t,:,:})$$

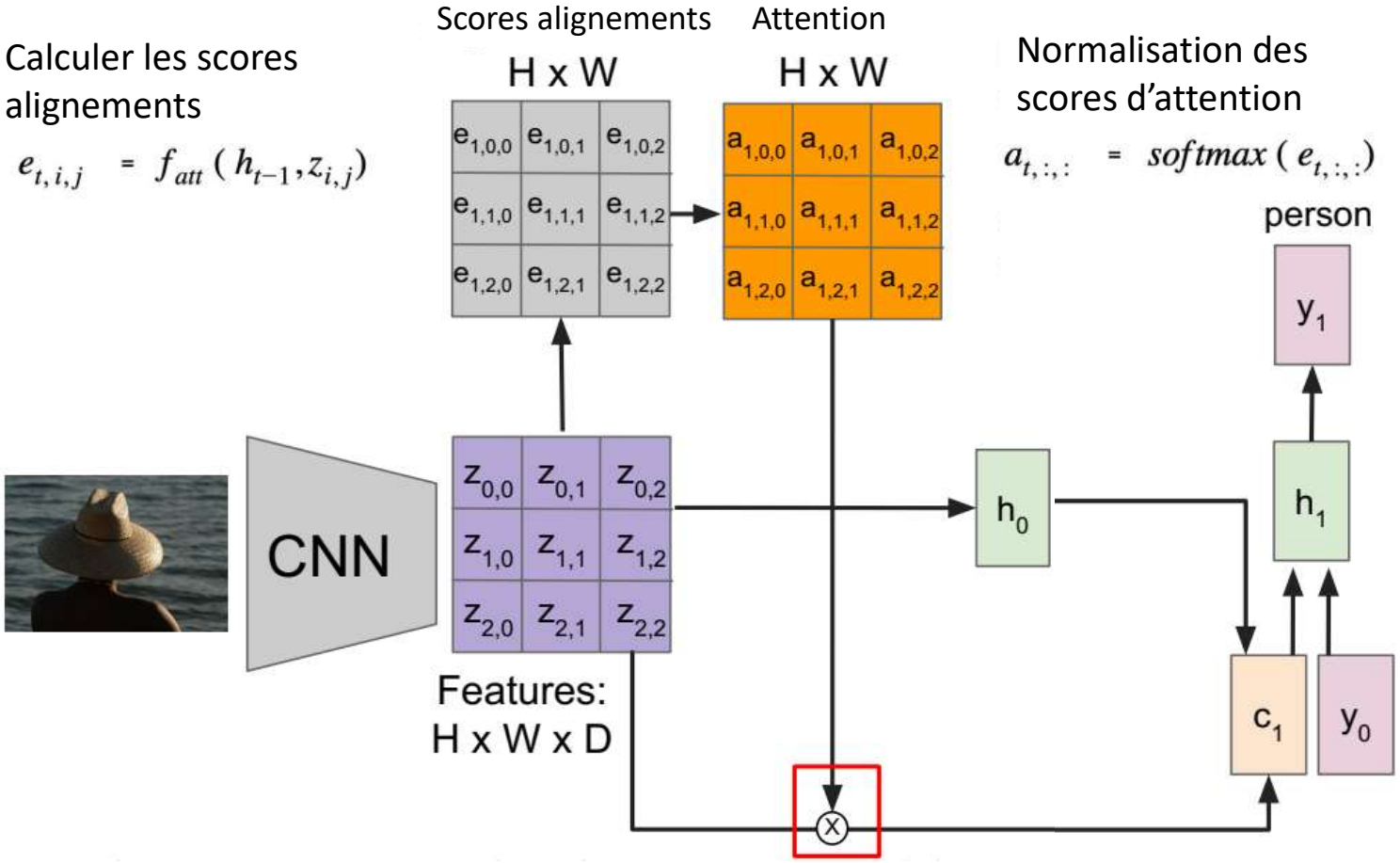
Calculer le vecteur de contexte

$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Image Captioning avec attention

Calculer les scores alignements

$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$



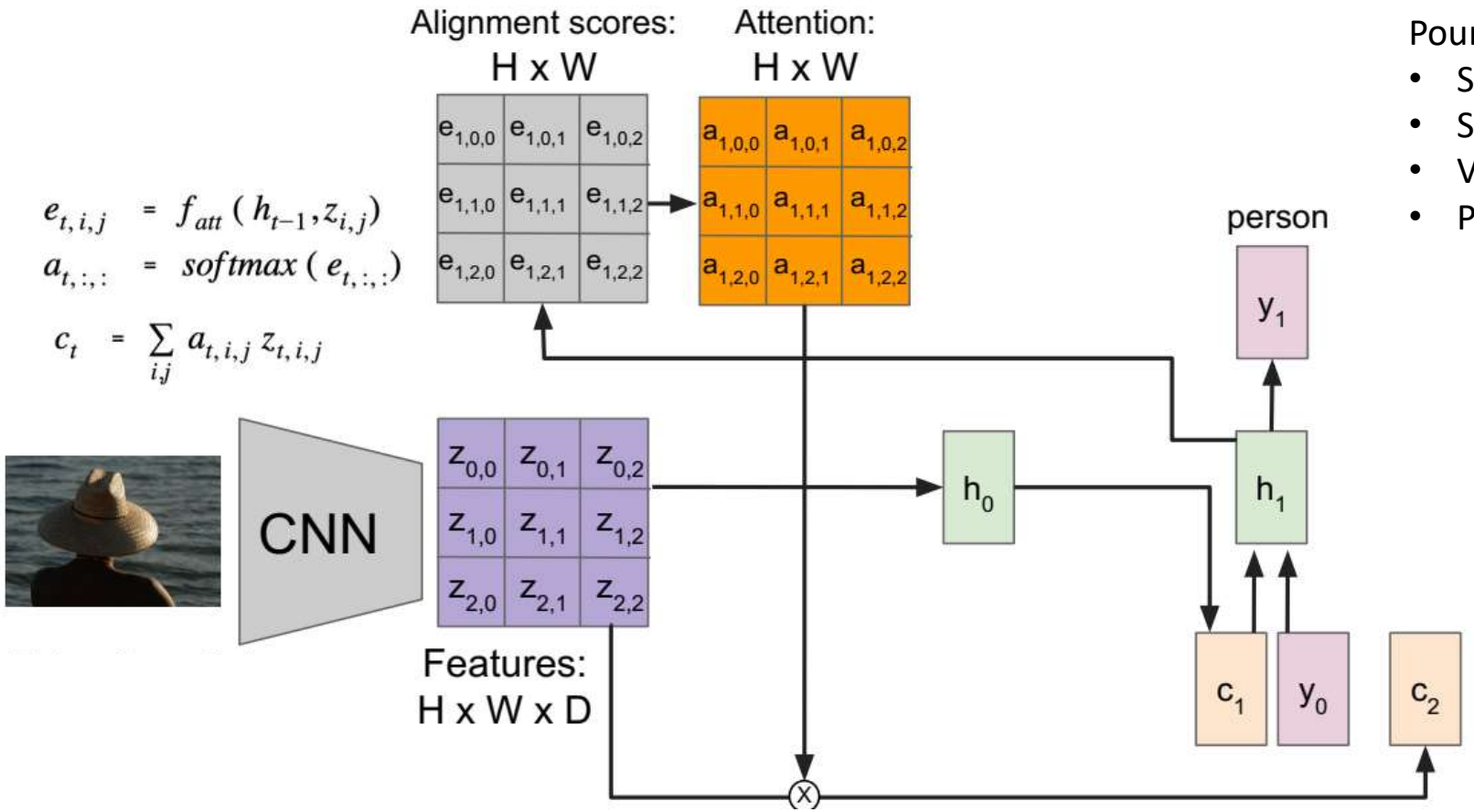
Calculer le vecteur de contexte

$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Génération du texte pour ce vecteur de contexte

Un vecteur de contexte spécifique a chaque étape

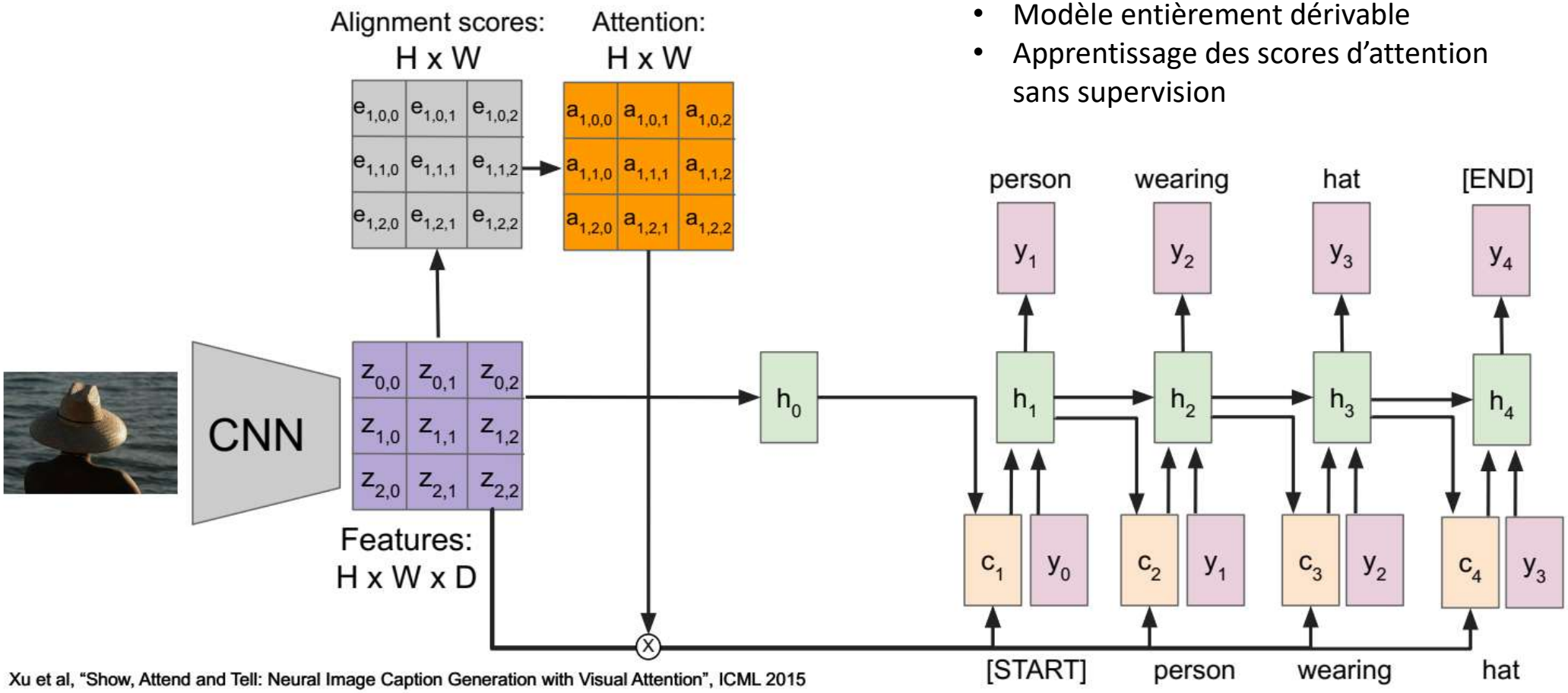
Image Captioning avec attention



Pour chaque étape, calcul :

- Scores d'alignement
- Score d'attention
- Vecteur de contexte
- Prédiction

Image Captioning avec attention

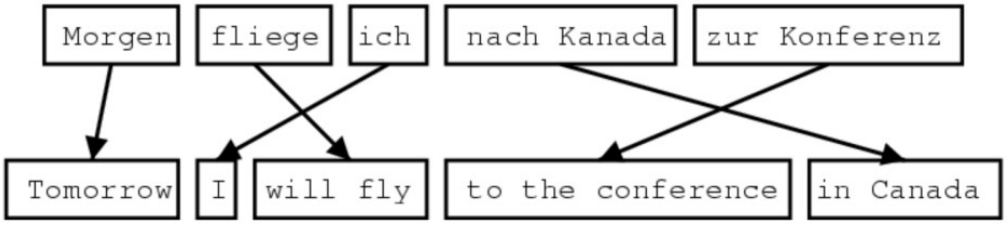


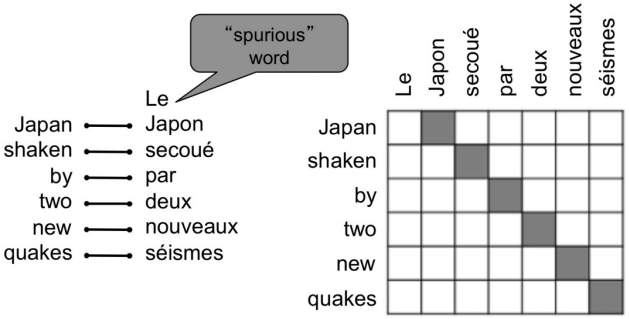
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

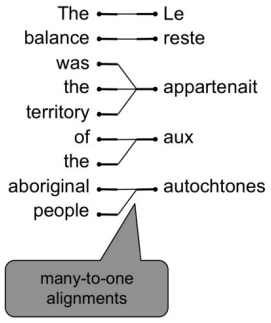
- Modèle entièrement dérivable
- Apprentissage des scores d'attention sans supervision

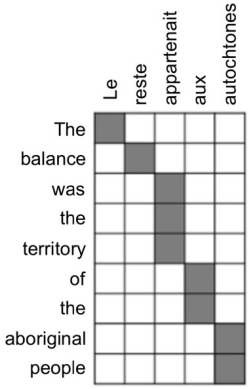
Traduction automatique

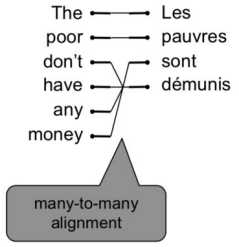
- Problème d'alignement

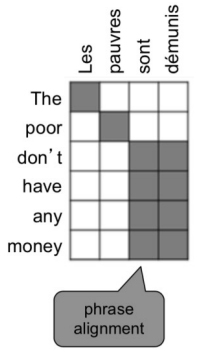








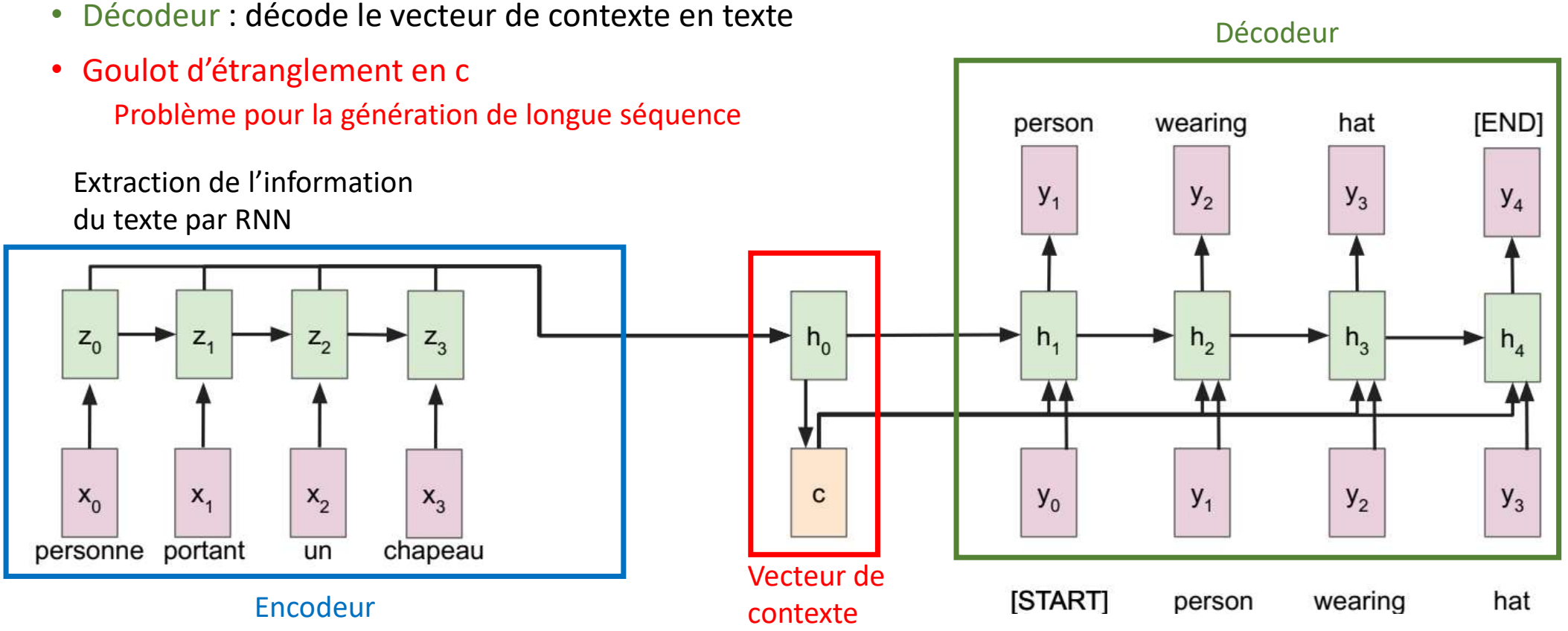




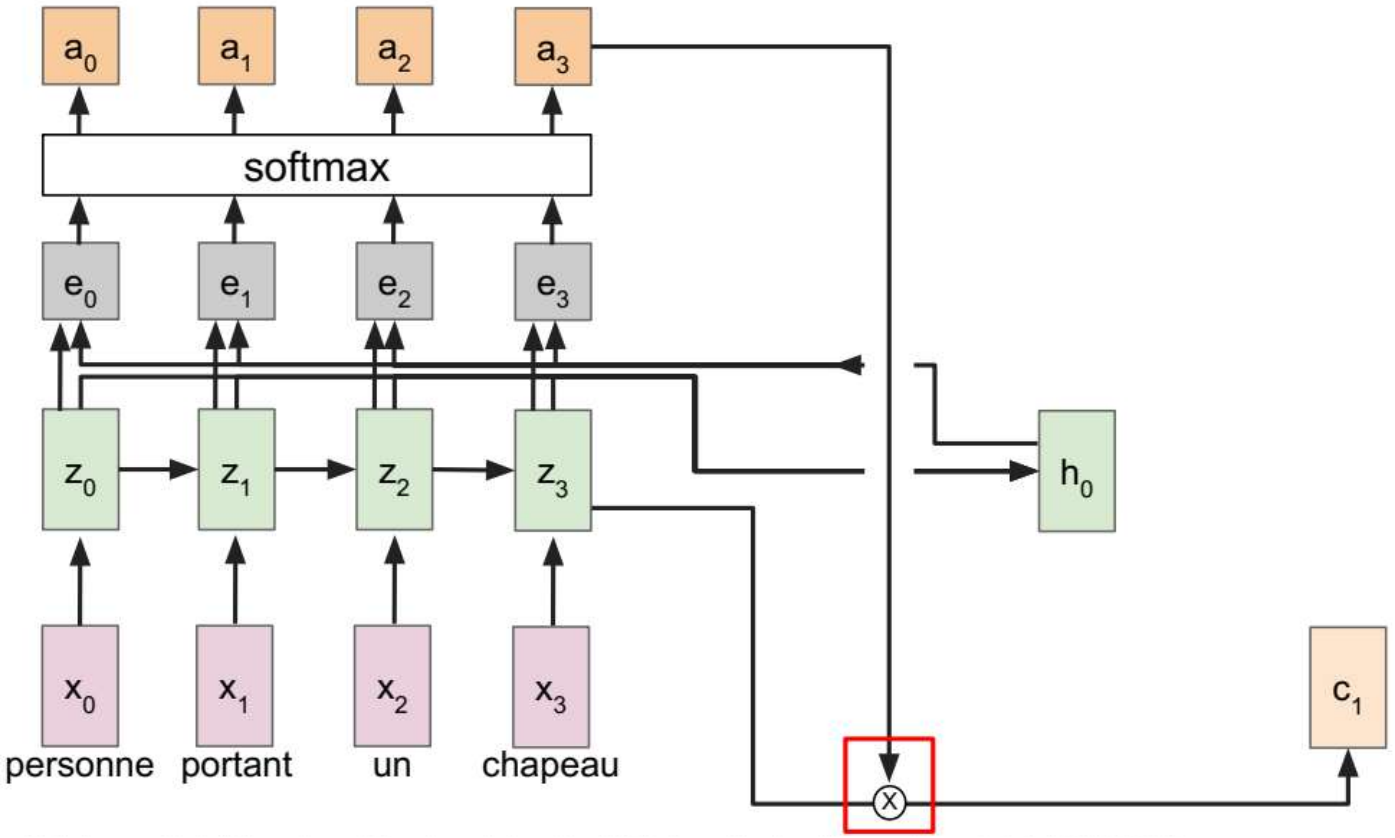
Traduction automatique

- Encodeur : encode l'information de l'image dans un vecteur de contexte
- Décodeur : décode le vecteur de contexte en texte
- Goulot d'étranglement en c

  Problème pour la génération de longue séquence



Traduction automatique avec attention



Calcul des scores d'alignement

$$e_{t,i} = f_{att}(h_{t-1}, z_i)$$

$f_{att}(.)$ is an MLP

Normalisation des scores d'attention

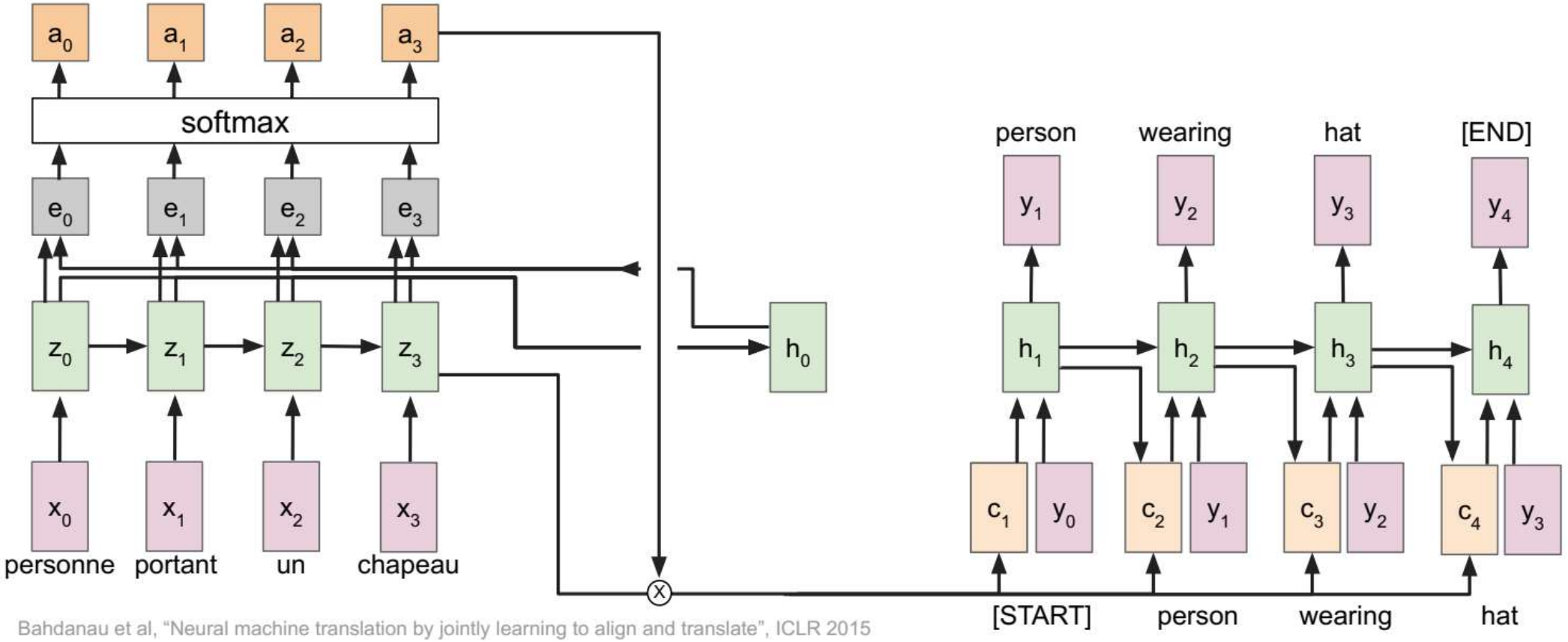
$$a_{t,:} = softmax(e_{t,:})$$

$0 < a_{t,i,j} < 1,$

Calcul du vecteur de contexte

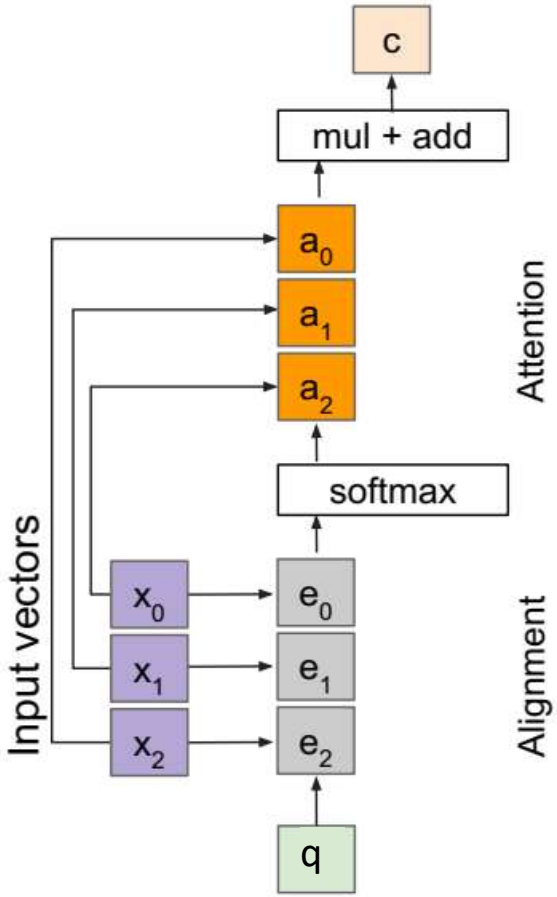
$$c_t = \sum_i a_{t,i} z_{t,i}$$

Traduction automatique avec attention



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Mécanisme d'attention



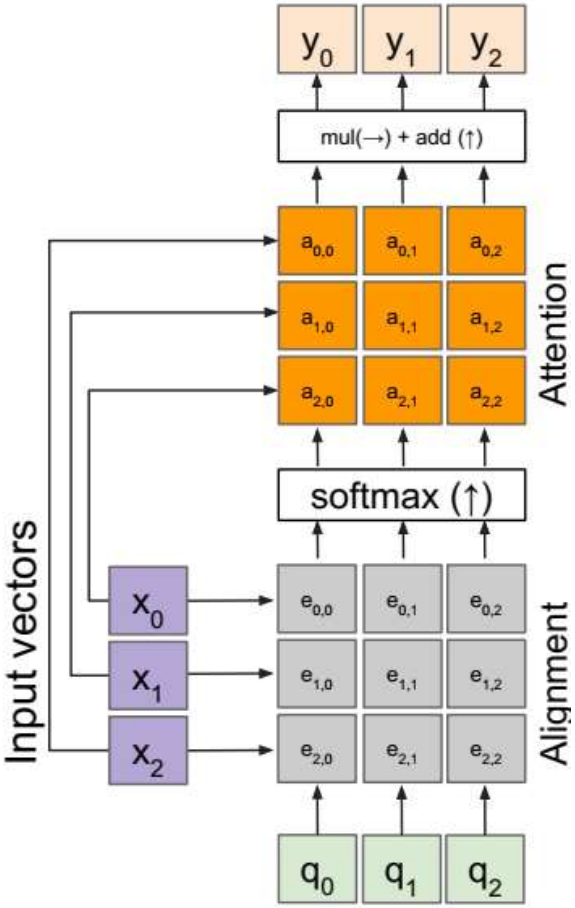
Sorties:
Vecteur de contexte: c (dim:D)

Operations:
Alignement: $e_i = f_{att}(q, x)$
Attention: $a = softmax(e)$
Sortie: $c = \sum_i \ a_i x_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
Query: q (dim:D)

Invariant à la permutation
L'ordre n est pas pris en compte

Mécanisme d'attention



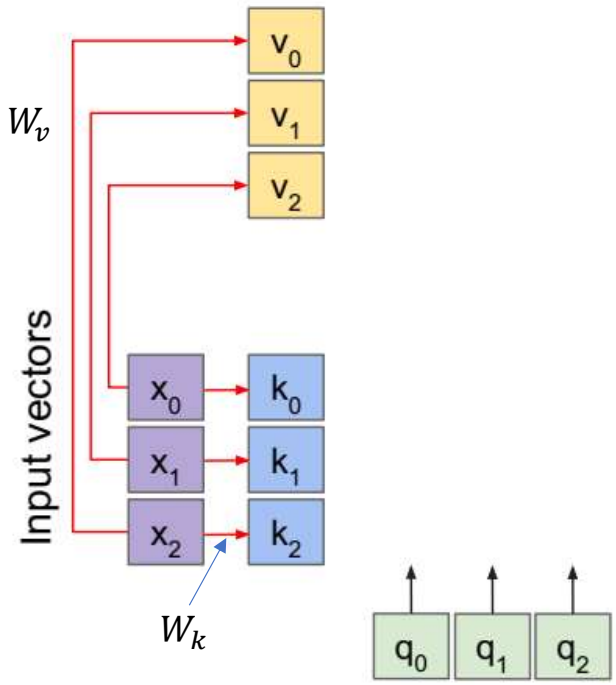
Sorties:
Vecteur de contexte: y (dim:D)

Operations:
Alignement: $e_i = f_{att}(q, x)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i \ a_{ij} x_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
Query: q (dim: MxD)

Invariant à la permutation
L'ordre n est pas pris en compte

Mécanisme d'attention



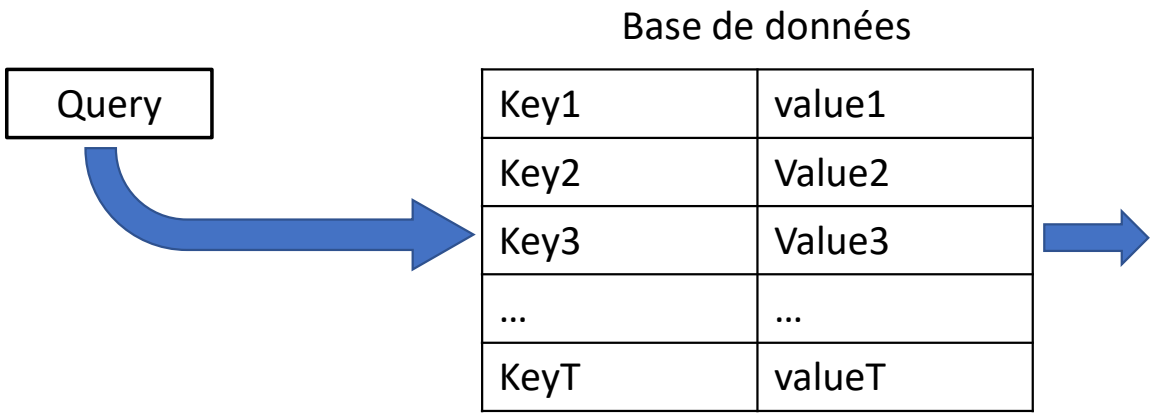
Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$

Entrées:
Vecteur d'entrée: x (dim:NxD)
Query: q (dim: MxD)

Invariant à la permutation
L'ordre n est pas pris en compte

Mécanisme d'attention

Inspiration : rechercher une value v_i pour une query q à partir d'un key k_i dans une base de données

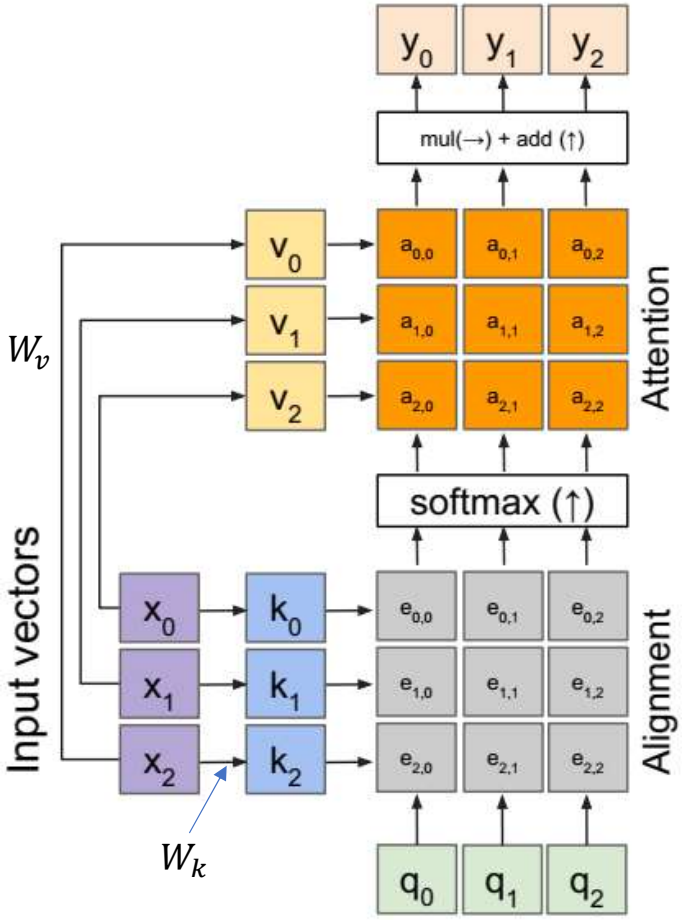


$$attention(q, k, v) = \sum_i similarity(q, k_i) \times v_i$$

Exemple : traduction automatique

- Query : q vecteur caché d'entré
- Key : k_i vecteur du i-ème mot d'entrée
- Value :$v_i$vecteur du i-ème mot d'entrée

Mécanisme d'attention



Sorties:
Vecteur de contexte: c (dim:D_v)

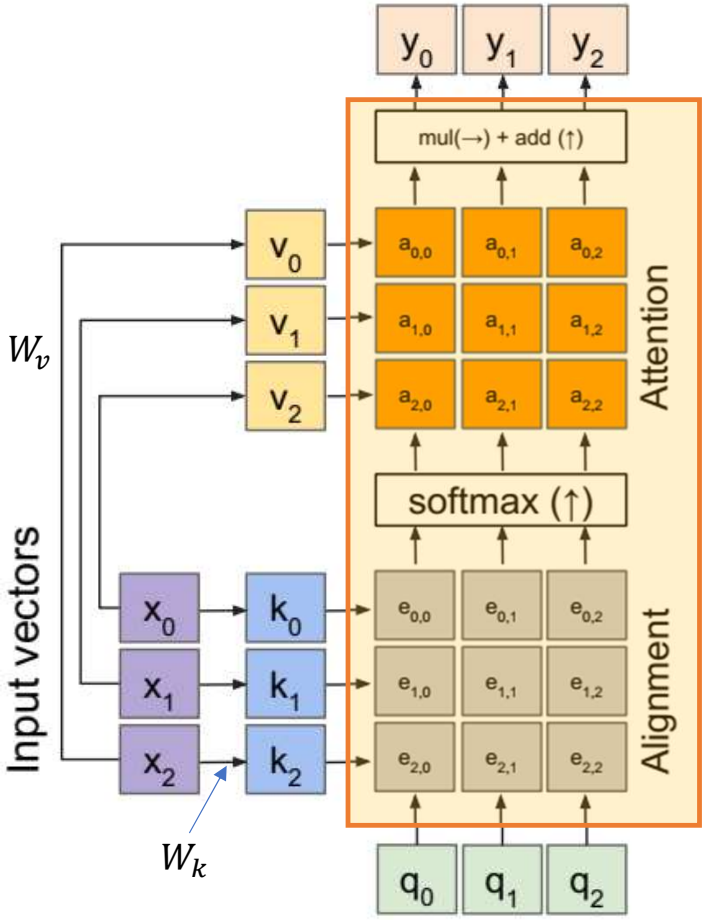
Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$
Alignement: $e_{i,j} = f_{att}(q_j, k_i)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i a_{i,j} v_i$

Calcul de l'alignement
- $e_{i,j} = q_j . k_i$
- $e_{i,j} = q_j . k_i / \sqrt{D}$
- $e_{i,j} = q_j W_{att} k_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
Query: q (dim: MxD_k)

Invariant à la permutation
L'ordre n est pas pris en compte

Mécanisme d'attention

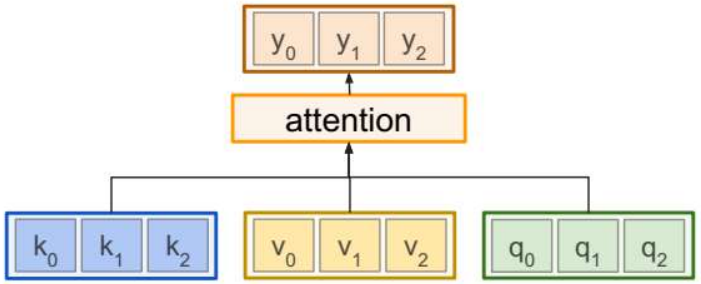


Sorties:
Vecteur de contexte: c (dim:D_v)

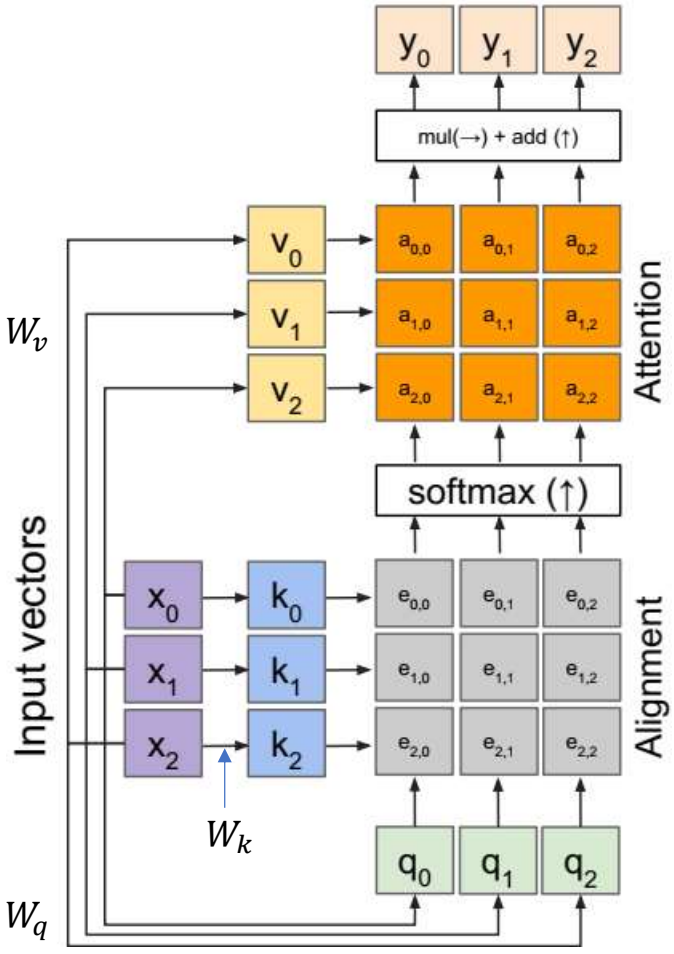
Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$
Alignement: $e_{i,j} = f_{att}(q_j, k_i)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i a_{i,j} v_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
Query: q (dim: MxD_k)

Attention layer



Self attention



Sorties:
Vecteur de contexte: c (dim:D_v)

Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$
Query vector: $q = xW_q$
Alignement: $e_{i,j} = f_{att}(q_j, k_i)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i \ a_{i,j} v_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
~~Query: q (dim: MxD$_k$)~~

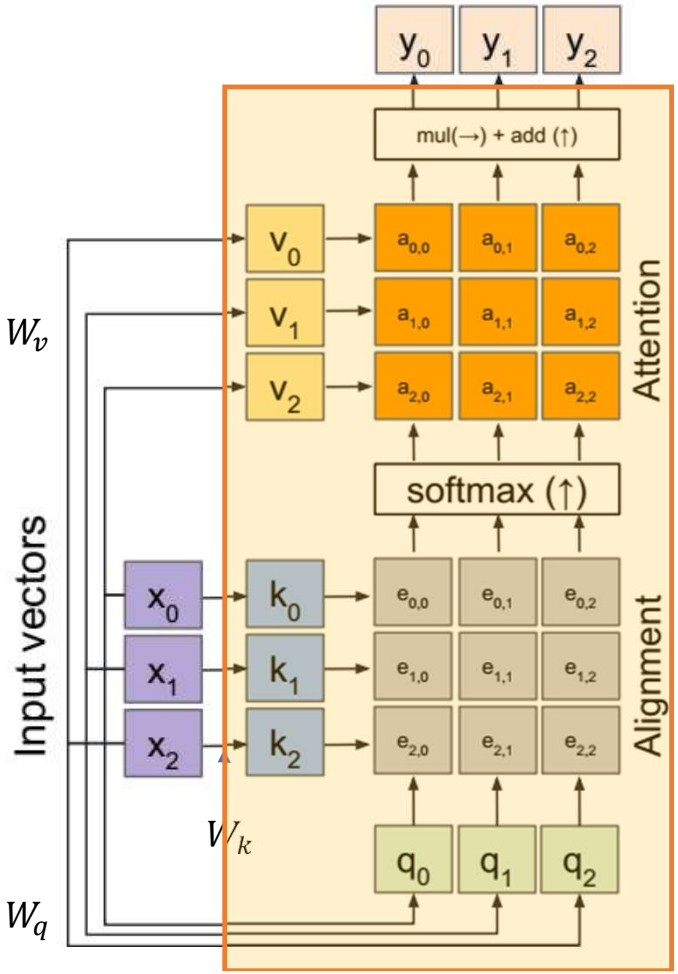
Calcul d'une attention entre chaque élément d'une séquence.

Les vecteurs query sont calculés depuis l'entrée.

Calcul de l'alignement
- $e_{i,j} = q_j . k_i$
- $e_{i,j} = q_j . k_i / \sqrt{D}$
- $e_{i,j} = q_j W_{att} k_i$

Invariant à la permutation
L'ordre n est pas pris en compte

Self attention



Sorties:
Vecteur de contexte: c (dim:D_v)

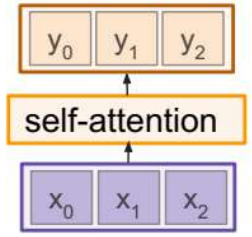
Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$
Query vector: $q = xW_q$
Alignement: $e_{i,j} = f_{att}(q_j, k_i)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i \ a_{i,j} v_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)
~~Query: q (dim: M×D_k)~~

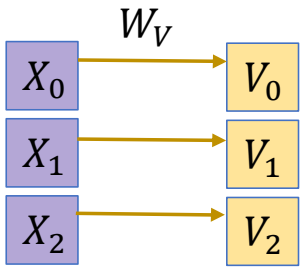
Calcul d'une attention entre chaque élément d'une séquence.

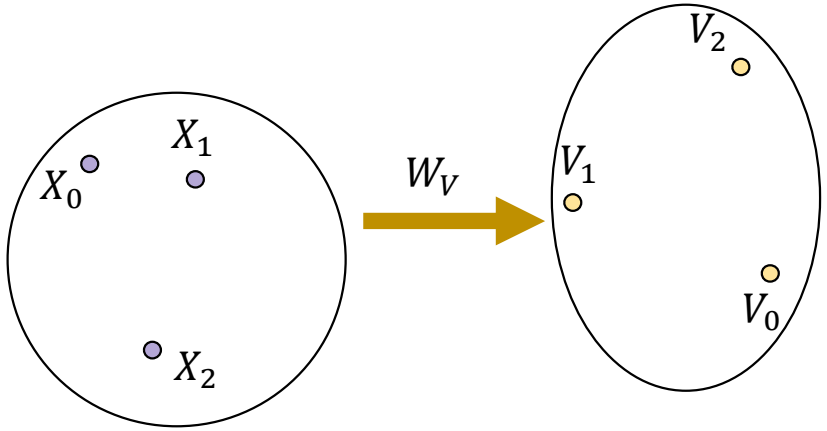
Les vecteurs query sont calculés depuis l'entrée.

Self-attention layer

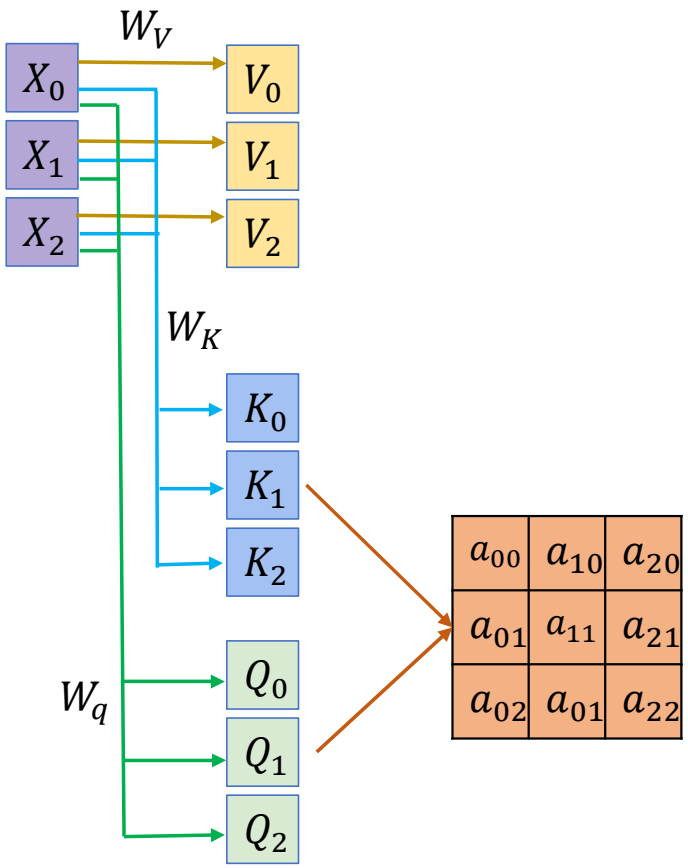


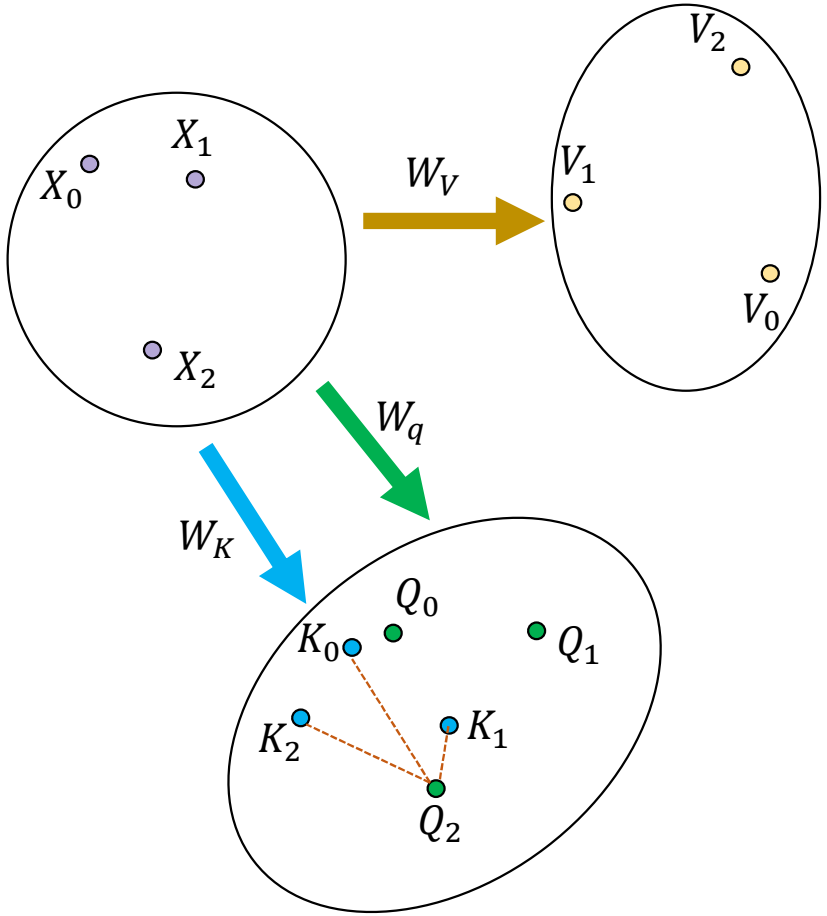
Self attention



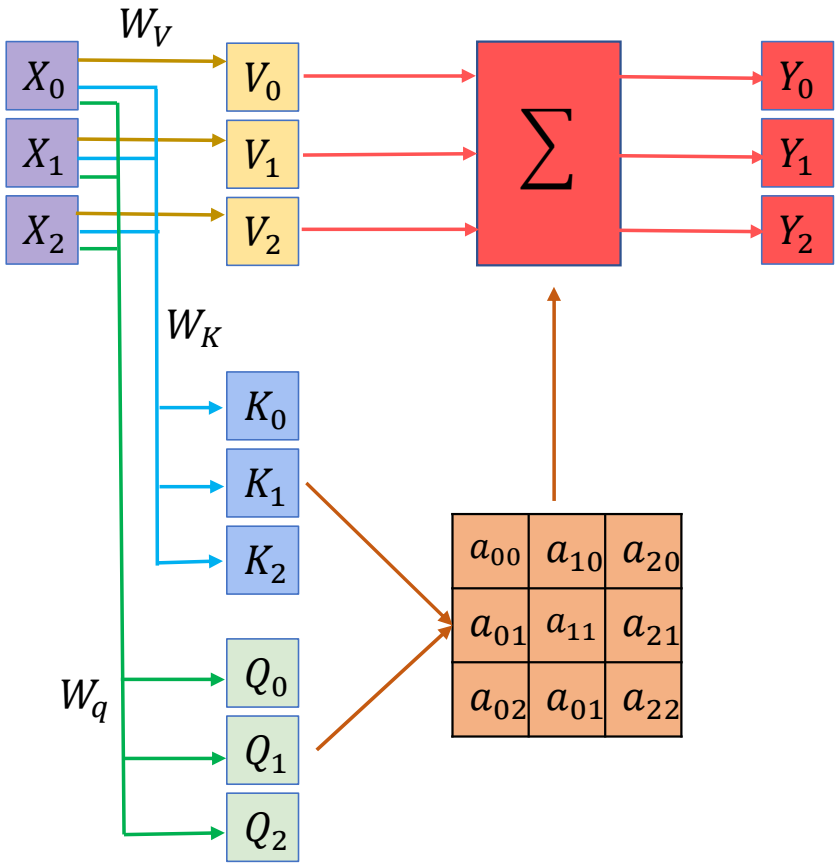


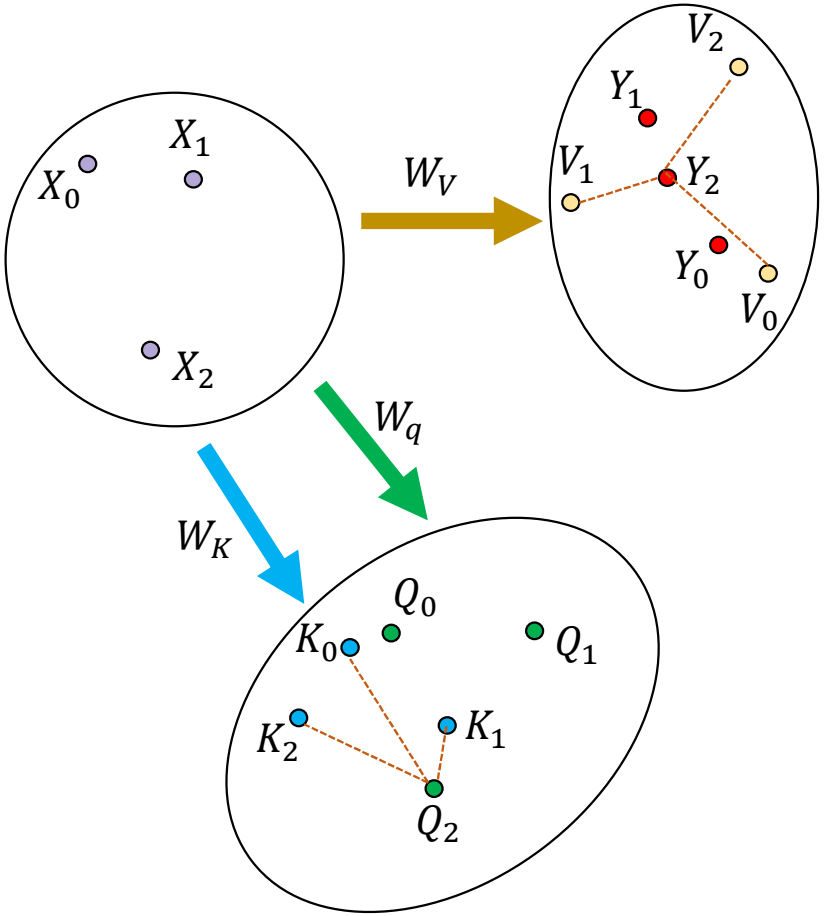
Self attention





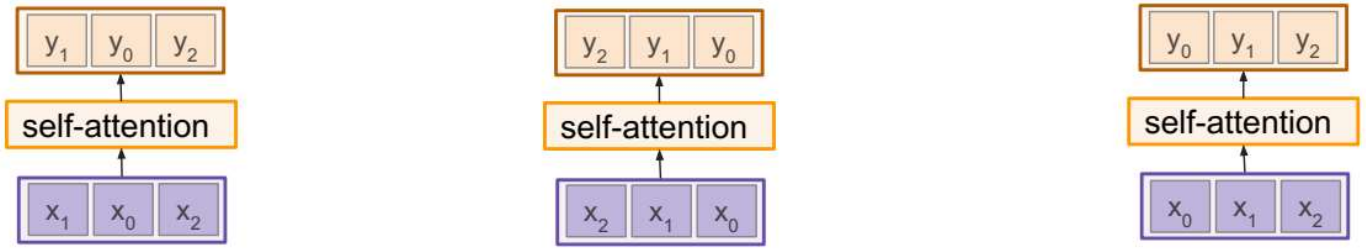
Self attention





Invariance à la permutation

- Les couches d'attention / self-attention sont invariants aux permutations des entrées
- Ne prends pas en compte l'ordre
- Comment encoder des séquences ?

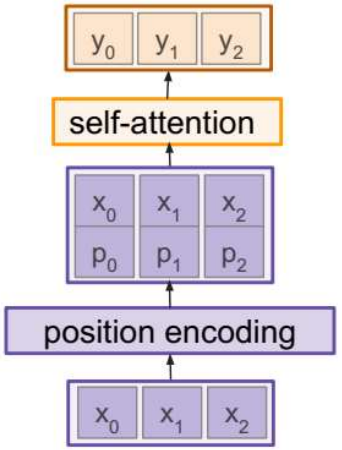


Encoder la position

- Utiliser une fonction $pos()$ qui encode la position de x_j dans une valeur p_j

$$\text{pos}: \text{N} \rightarrow R^d \qquad p_j = pos(j)$$

- Propriétés de $pos()$:
  - Fonction déterministique
  - Chaque position doit avoir un codage unique
  - Distance entre 2 positions doit être consistant avec des séquences de différente taille
  - Facilement généralisable à des séquences longues
    - Valeurs devraient être bornées



Encoder la position

- Assigner une valeur dans [0,1] dépendant de la position

  Chaque position doit avoir un codage unique

- Assigner une valeur linéairement croissante avec la position [1,2,3,…]

  Facilement généralisable à des séquences longues

- Intuition : Coder la position sur un vecteur

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

Coder sur des valeurs continues et non des bits

Chaque bit n a pas la même fréquence de changement

Encoder la position

- Vecteur de position

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

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

Dimension du vecteur de position

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

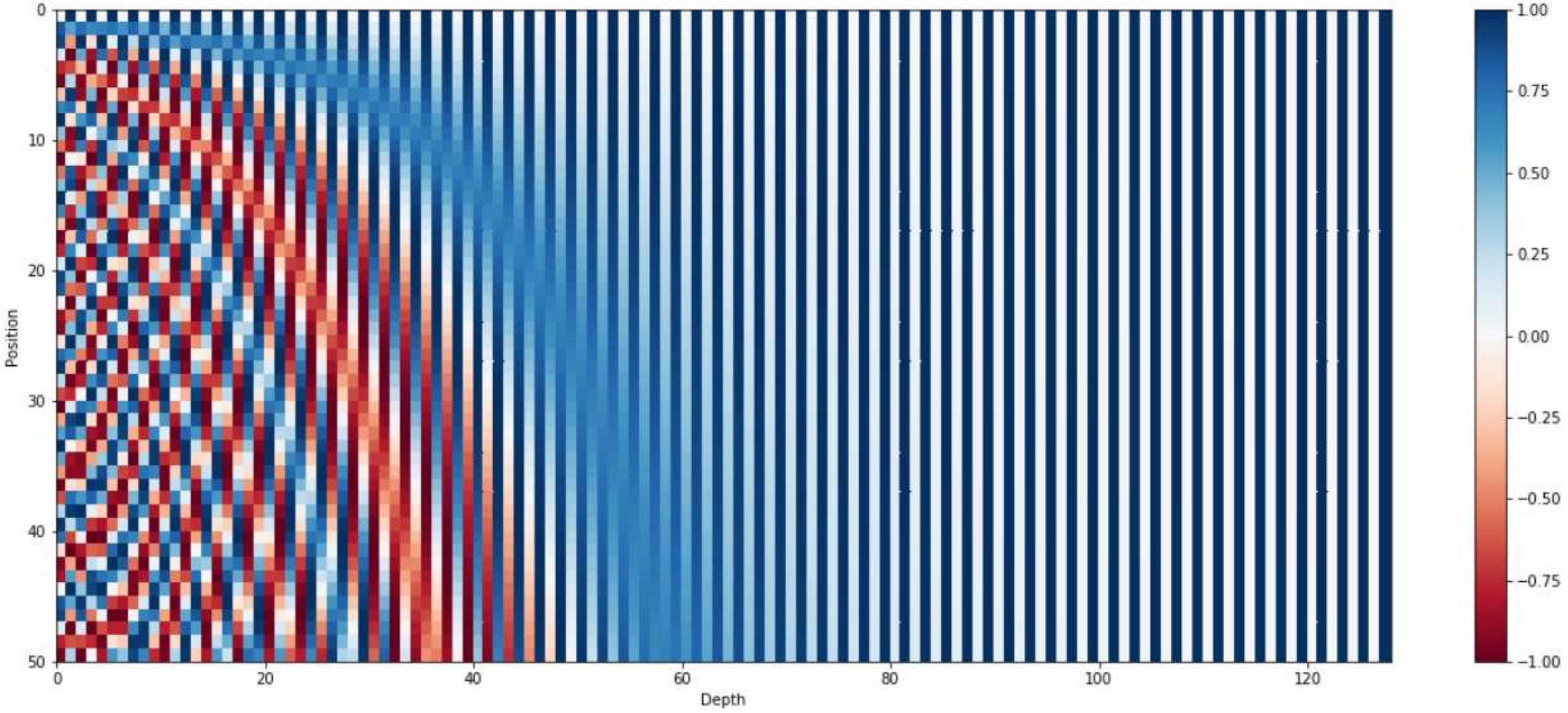
- Utiliser le vecteur de position

  Addition avec le vecteur de données

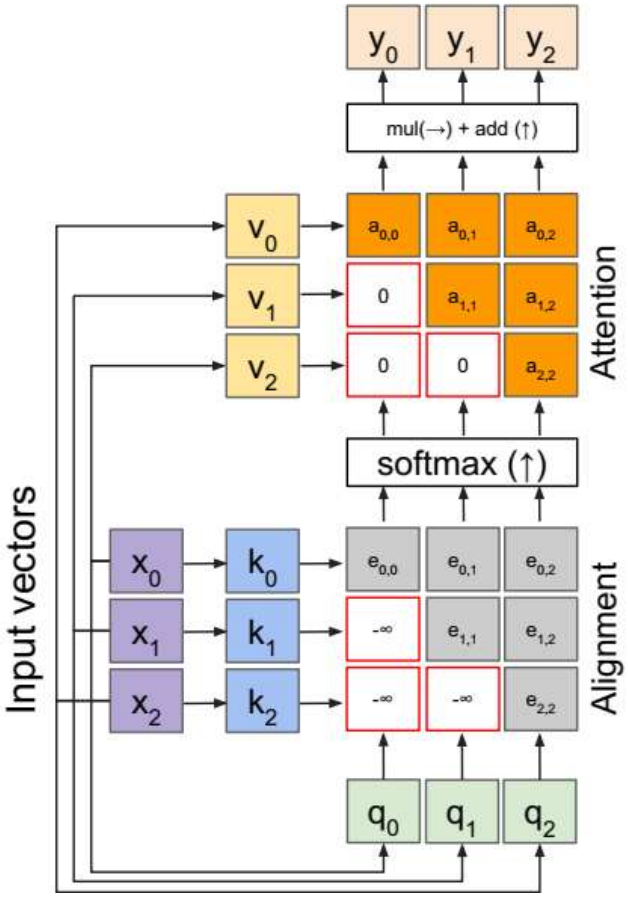
$$\psi'(w_t) = \psi(w_t) + \overrightarrow{p_t}$$

  Ce qui implique : $d_{\text{word embedding}} = d_{\text{postional embedding}}$

Encoder la position



Masked self-attention layer



Sorties:
Vecteur de contexte: c (dim:D_v)

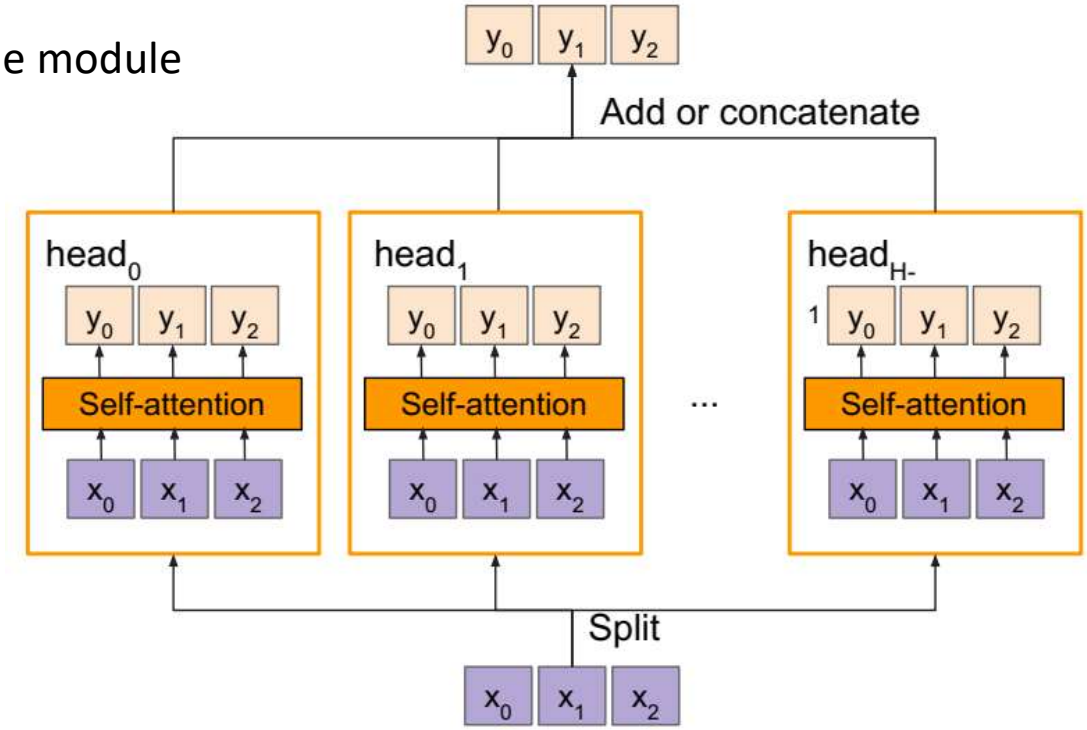
Operations:
Key vector: $k = xW_k$
Value vector: $v = xW_v$
Query vector: $q = xW_q$
Alignement: $e_{i,j} = f_{att}(q_j, k_i)$
Attention: $a = softmax(e)$
Sortie: $y_j = \sum_i \ a_{i,j} v_i$

Entrées:
Vecteur d'entrée: x (dim:NxD)

- Dans certain données séquentielle, on ne peut pas prédire à partir du futur
- Fixe les scores d'alignement à $-\infty$ pour les données futures

Multi-head attention layer

- Plusieurs modules d'attention en parallèle
- Chaque module à ses propres paramètres
- Agrégation de la sortie de chaque module



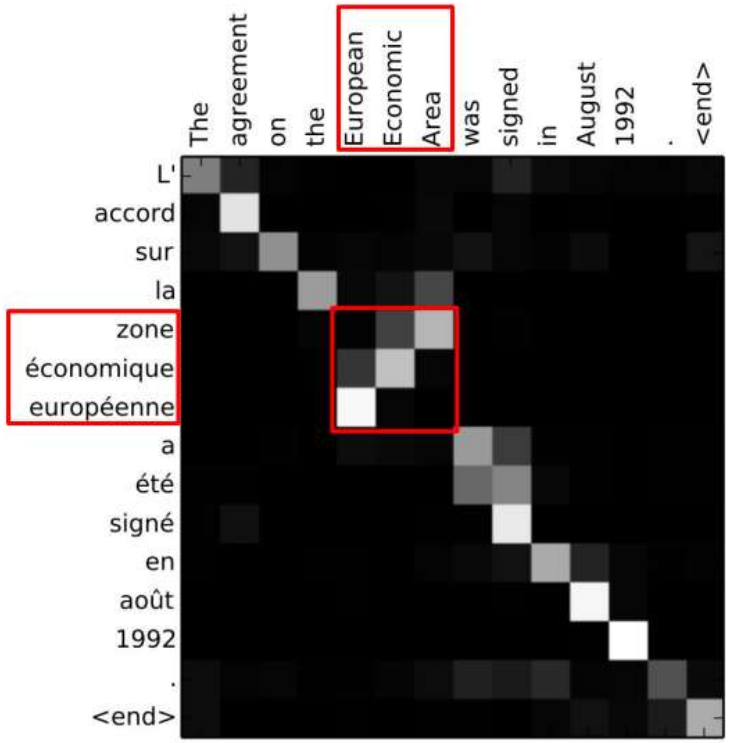
Visualisation de l’attention

Exemple de traduction du français vers l’anglais

Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L’accord sur la zone économique européenne a été signé en août 1992."

Le modèle apprend automatique l’ordre des mots et les aligne entre différents langages



Visualisation de l’attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Interprétation de l‘attention

- Explication des prédictions
- Validation du modèle
- Indentification de biais

Wrong

Baseline:
*A **man** sitting at a desk with a laptop computer.*

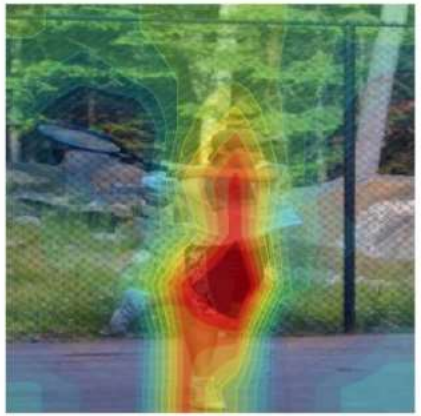
Right for the Right Reasons

Our Model:
*A **woman** sitting in front of a laptop computer.*

Right for the Wrong Reasons

Baseline:
*A **man** holding a tennis racquet on a tennis court.*

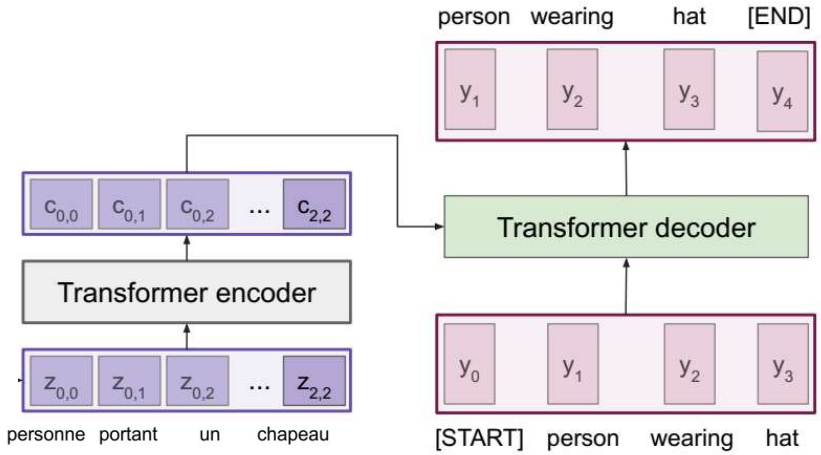
Right for the Right Reasons

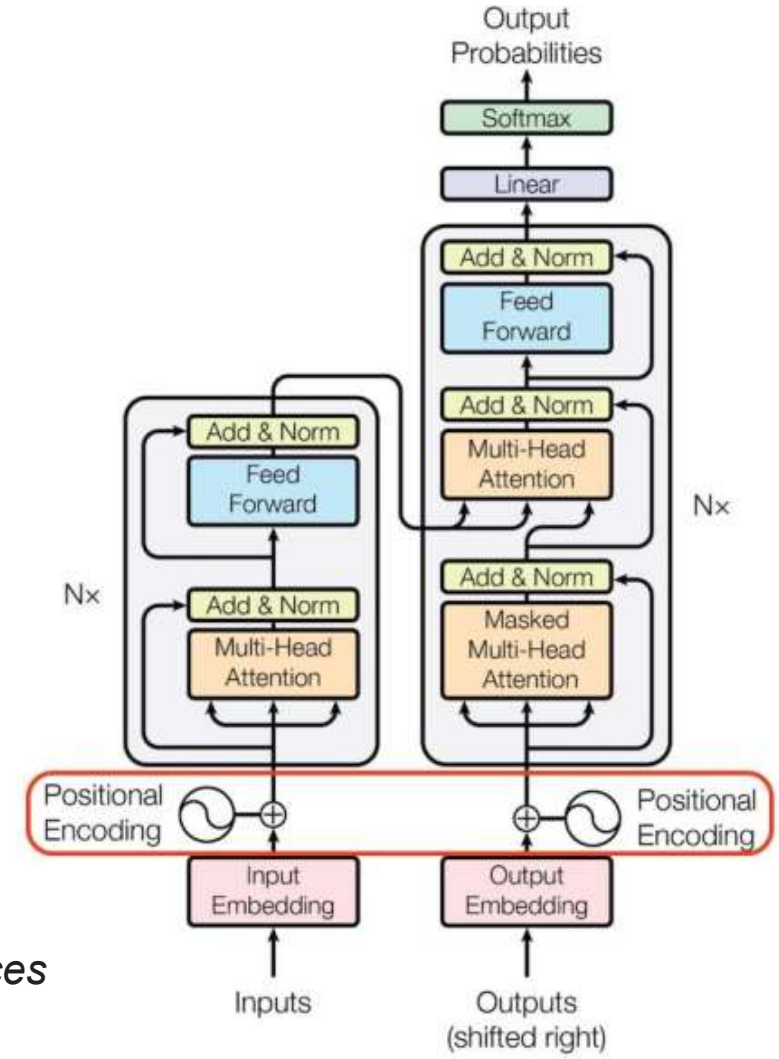


Our Model:
*A **man** holding a tennis racquet on a tennis court.*

Transformers

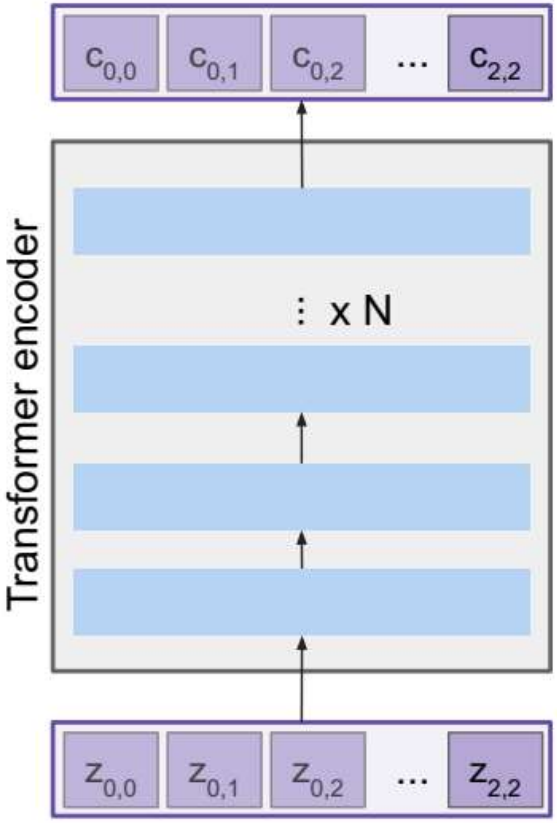
- Architecture encodeur - décodeur





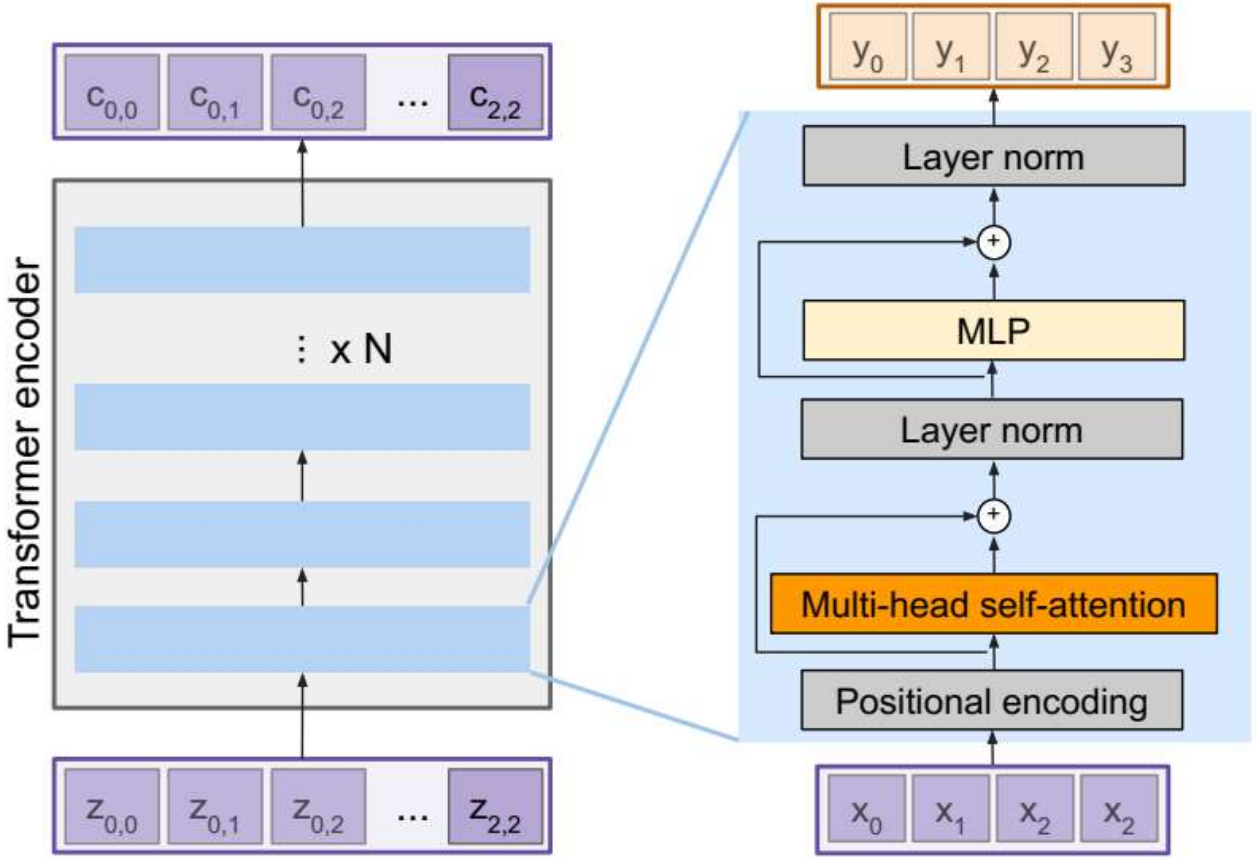
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer : encodeur



- Dans le modèle d'origine :
  - Nombre de blocs $N = 6$
  - Taille séquence $D_q = 512$

Transformer : encodeur

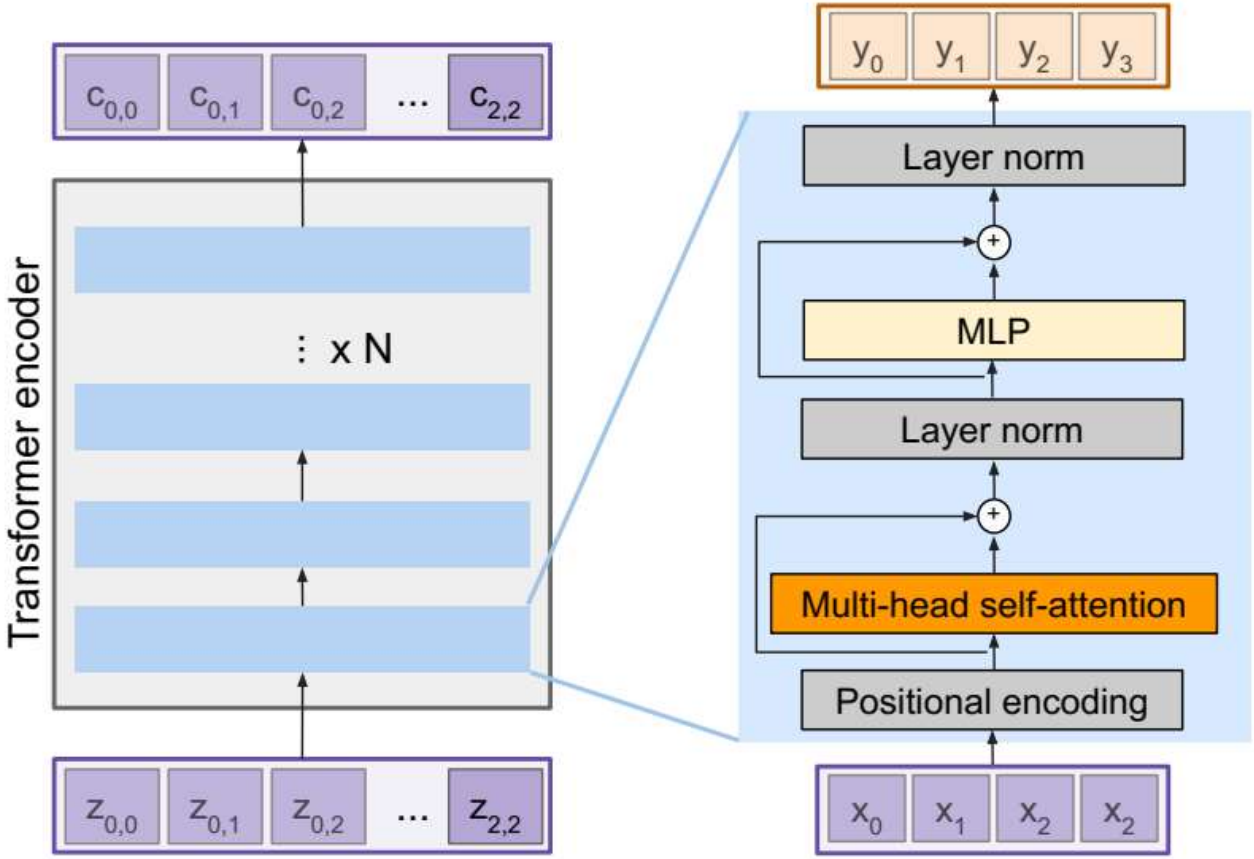


Connection résiduelle

Normalisation de chaque couche individuellement

MLP sur chaque vecteur individuellement

Transformer : encodeur



Entrée : ensemble de vecteur x

Sortie : ensemble de vecteur y

Interaction entre les vecteurs seulement par self-attention

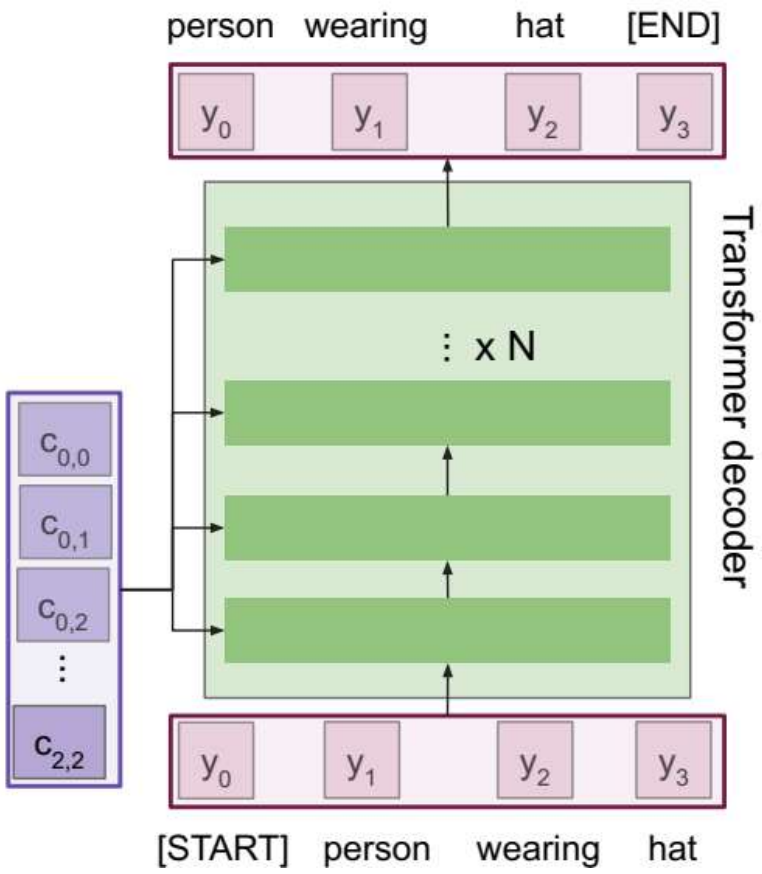
Normalisation et MLP traite chaque vecteur indépendamment

Longue séquence

Facilement parallélisable

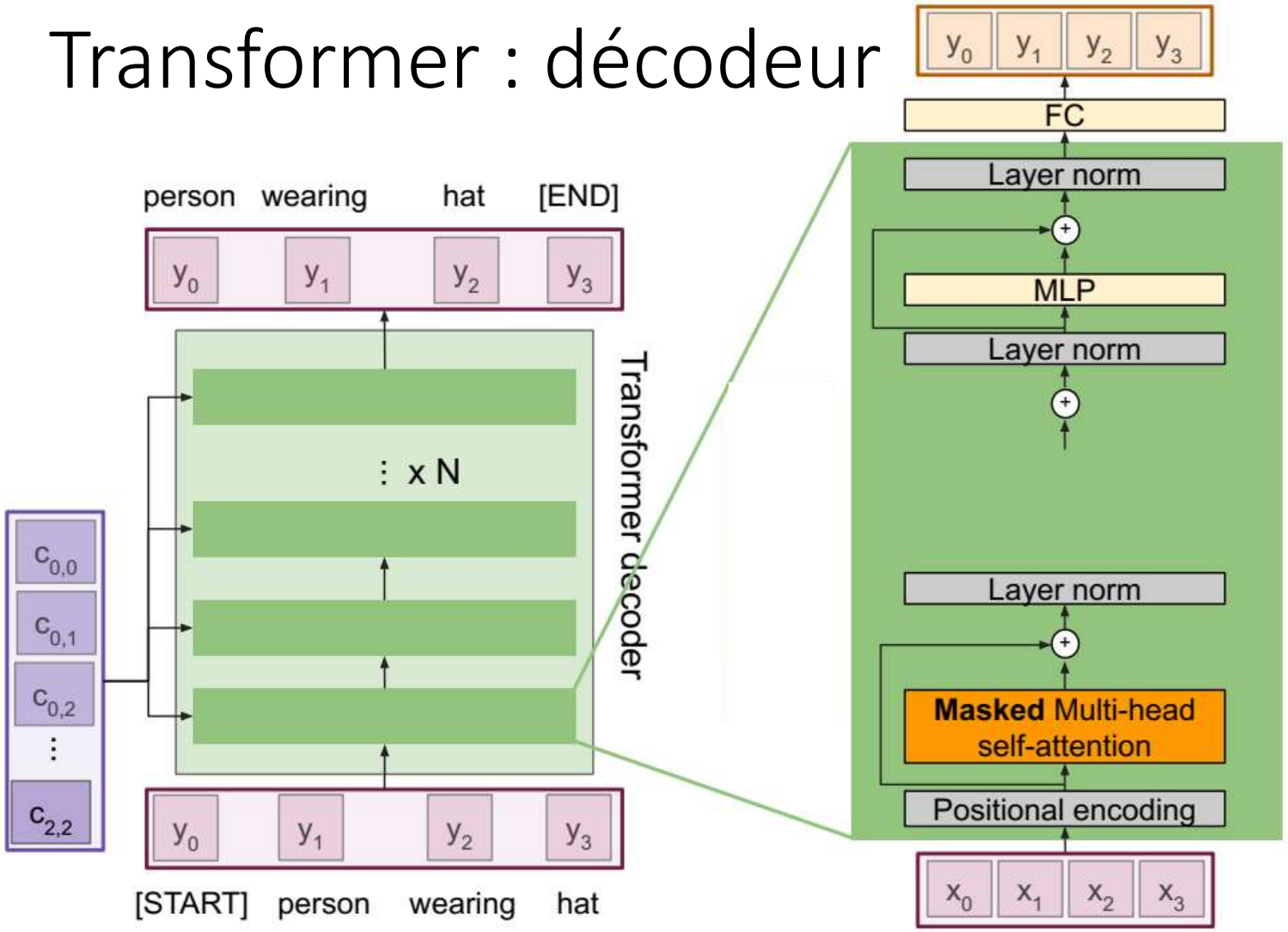
Demande beaucoup de mémoire

Transformer : décodeur



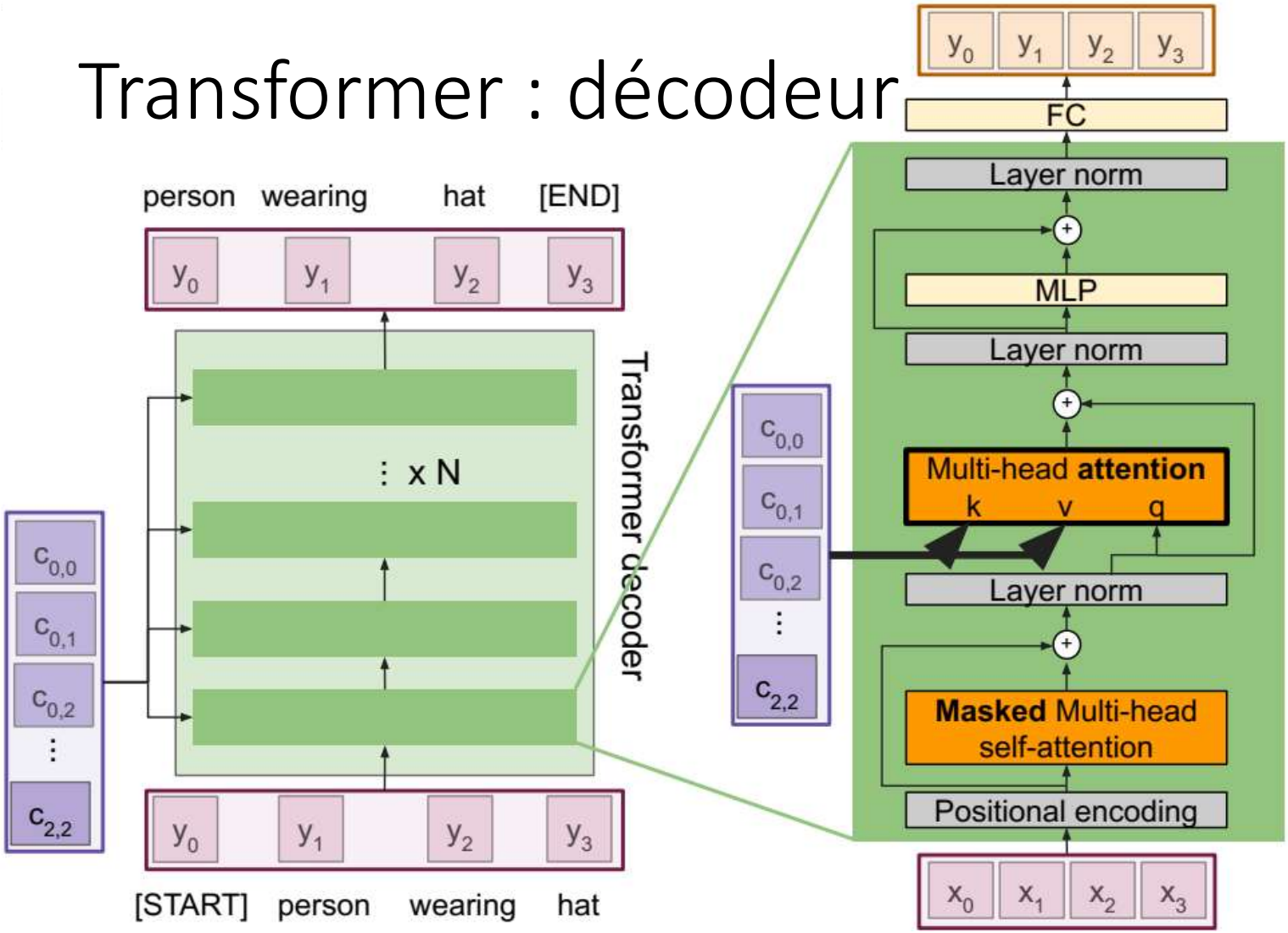
- Dans le modèle d'origine :
  - Nombre de blocs $N = 6$
  - Taille séquence $D_q = 512$

Transformer : décodeur



Identique à l'encodeur pour le moment

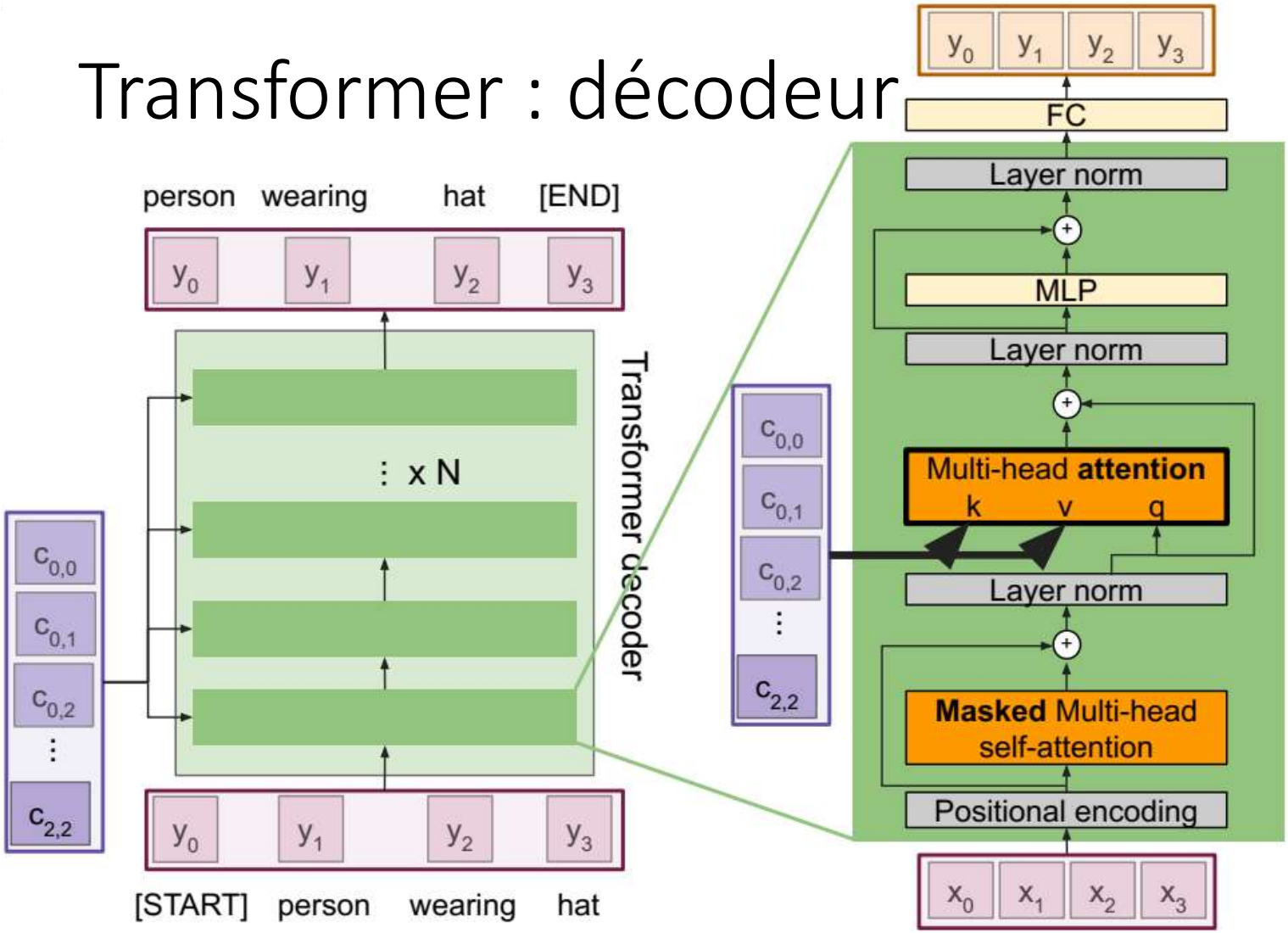
Transformer : décodeur



Multi-head attention

Injecter la sortie de l'encodeur dans chaque bloc du décodeur

Transformer : décodeur



- Entrée :
  ensemble de vecteur x
  ensemble de vecteur de contexte c
- Sortie : ensemble de vecteur y

- Masked self-attention: interaction uniquement avec le passé
- Attention : interaction encodeur - décodeur

- Longue séquence
- Facilement parallélisable
- Demande beaucoup de mémoire

Traitement de données séquentielles

RNN

- Dépendance de long terme avec les LSTM
- Disparition / explosion du gradient
- Grand nombre d'epochs nécessaire
- Récurrence pose de problème de parallélisation

Réseaux transformers

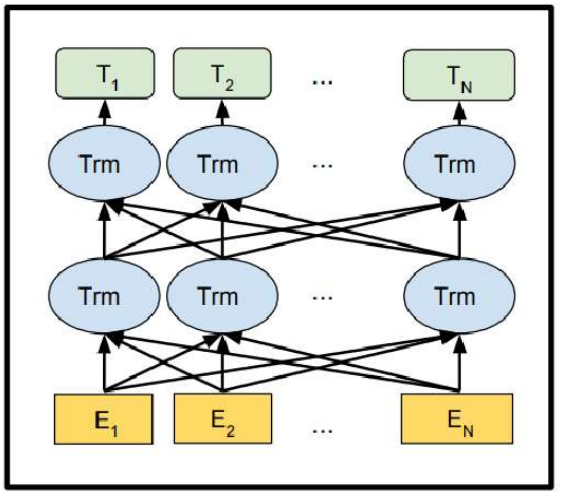
- Facilite les dépendance a long terme
- Pas de disparition / explosion du gradient
- Peut opérer sur de ensembles ordonnées ou non
- Pas de récurrence -> parallélisation
- Requière beaucoup de mémoire (matrice NxM pour chaque module d'attention)

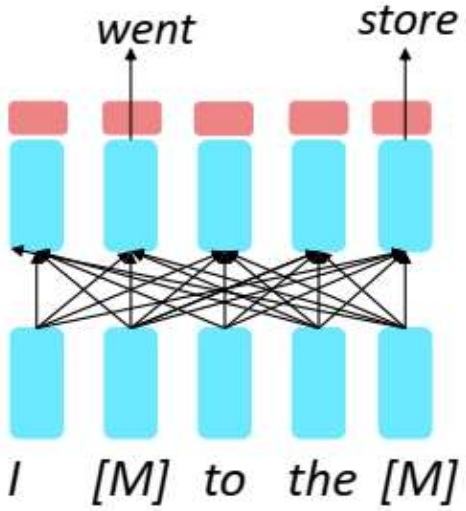
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n : taille sequence
d : dimension
k : taille du kernel
r : taille du voisinage

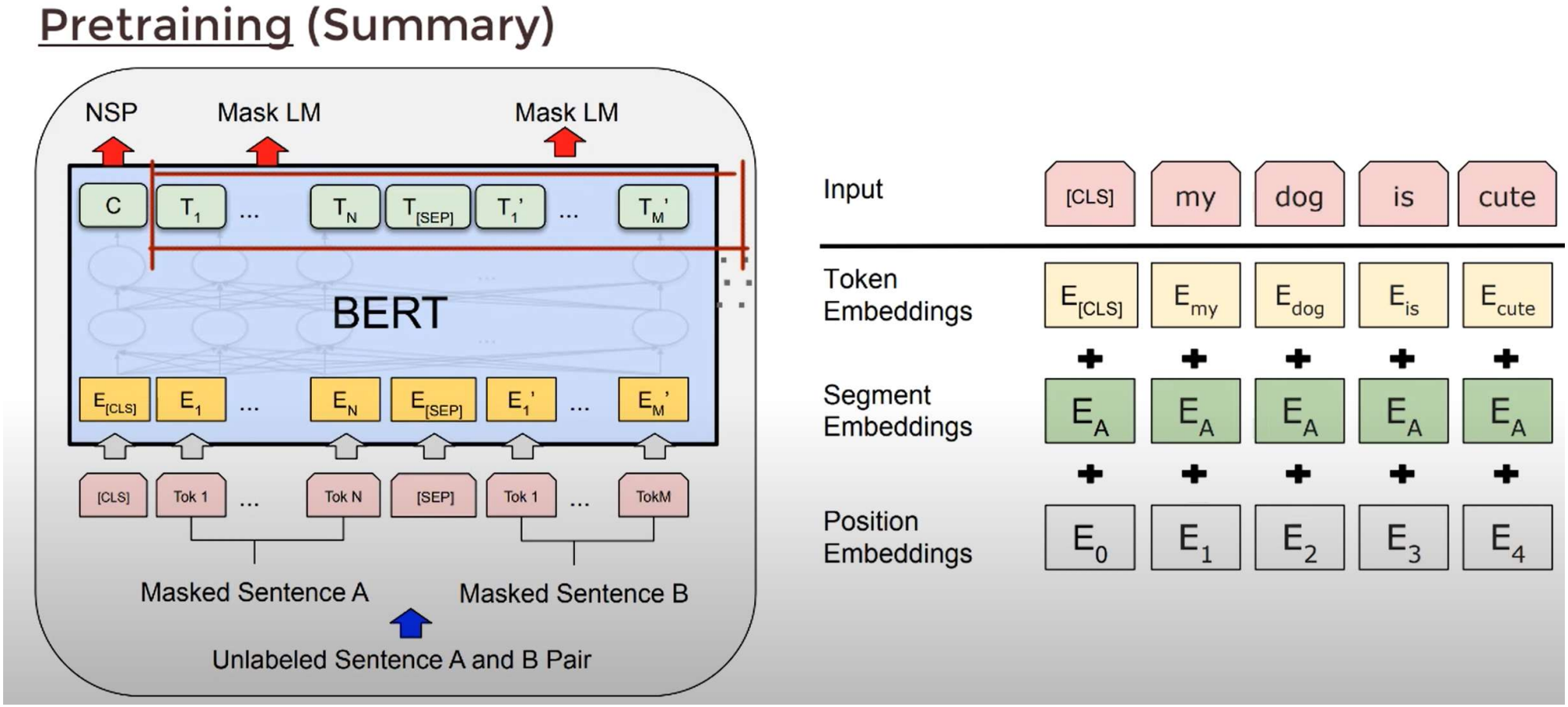
Birectional Encoder Representation from Transformers (BERT)

- Apprendre un codage du texte
  - Utiliser beaucoup de texte
  - Masquer aléatoirement des mots (15%)
  - Prédire les mots cachés
  - Encodage des mots dépends du contexte
  - Attention bidirectionelle
- Apprendre un codage des phrases
  - Prédire si deux phrases sont adjacentes
- Architecture
  - Blocs de d'encodeur (pas de décodeur) : 24
  - Neurones pas couche : 1024
  - Attention heads : 16
  - Taille de séquence : 512
- Nombreuses variantes : smallBERT, BioBERT, …

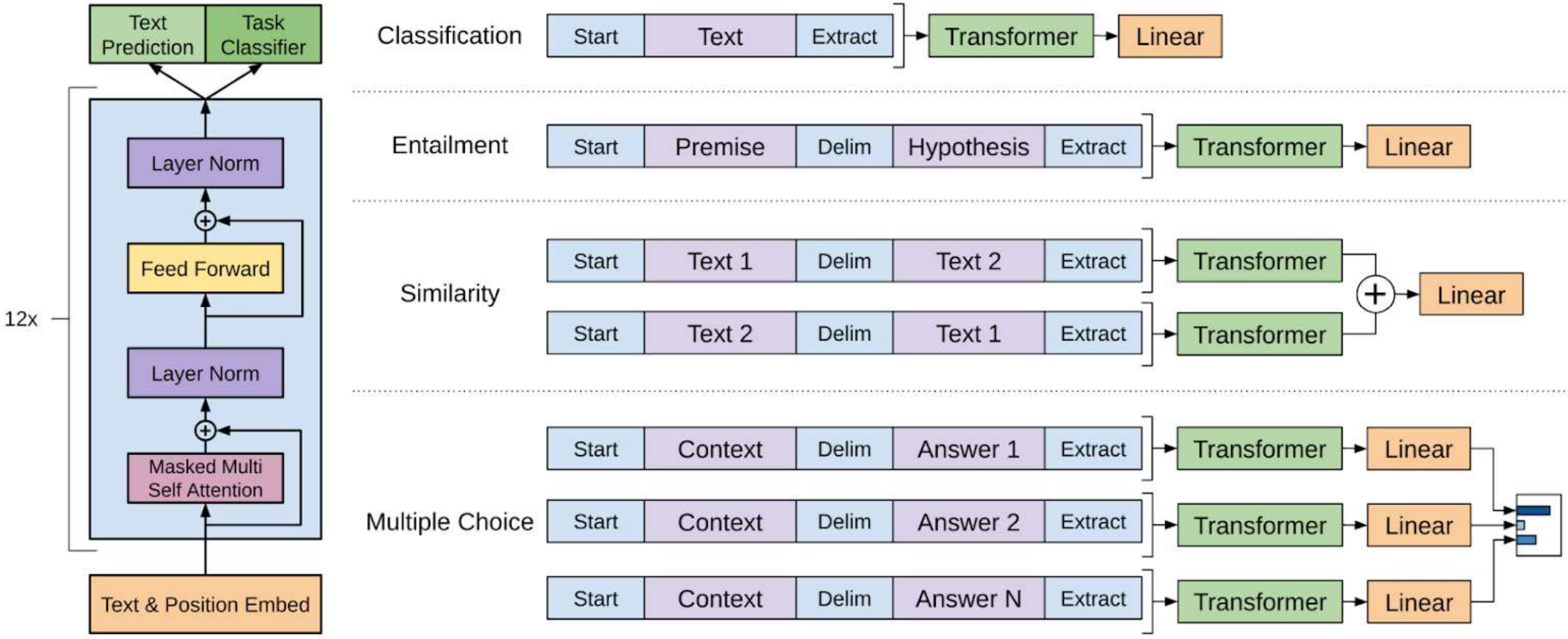


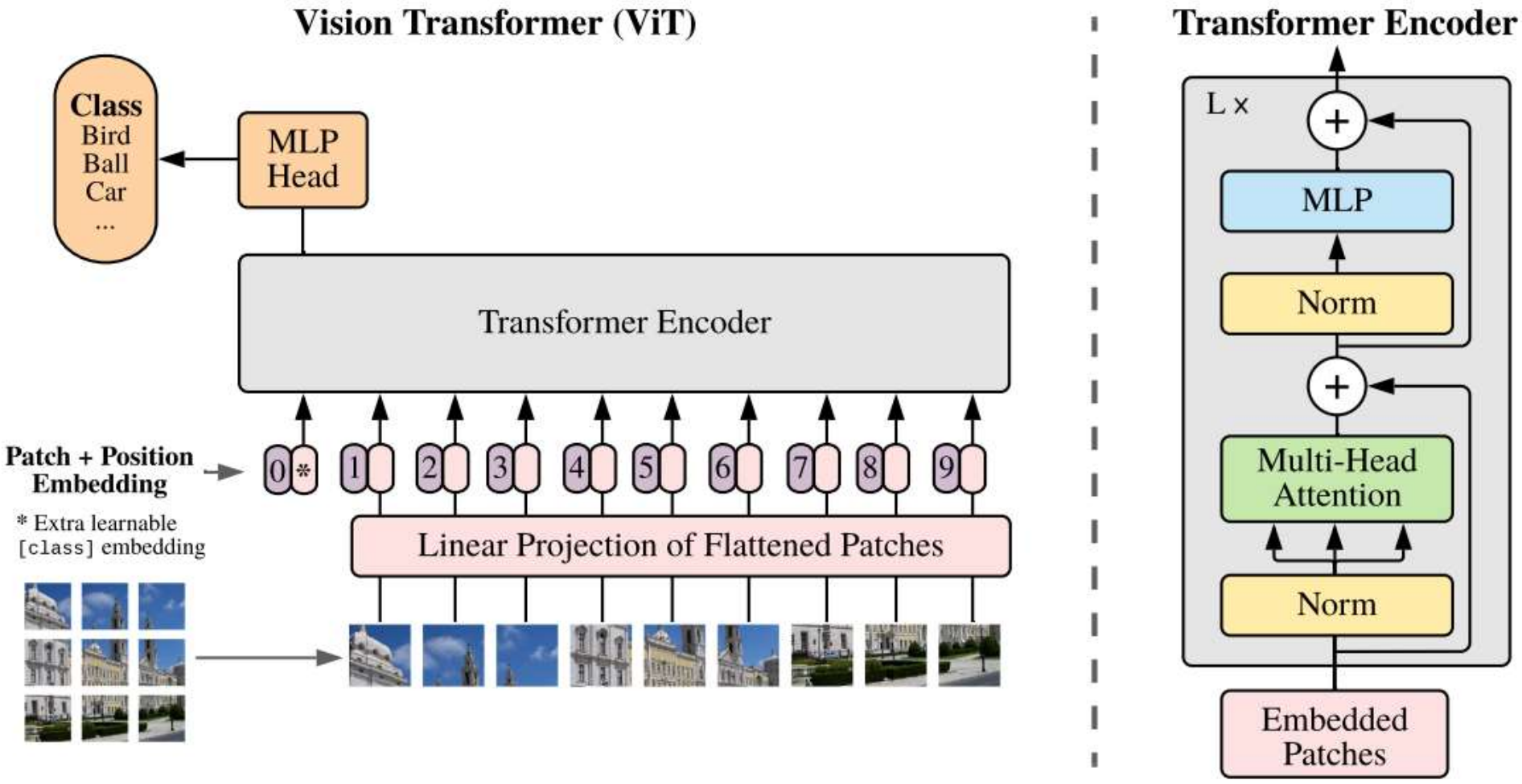


Birectional Encoder Representation from Transformers (BERT)



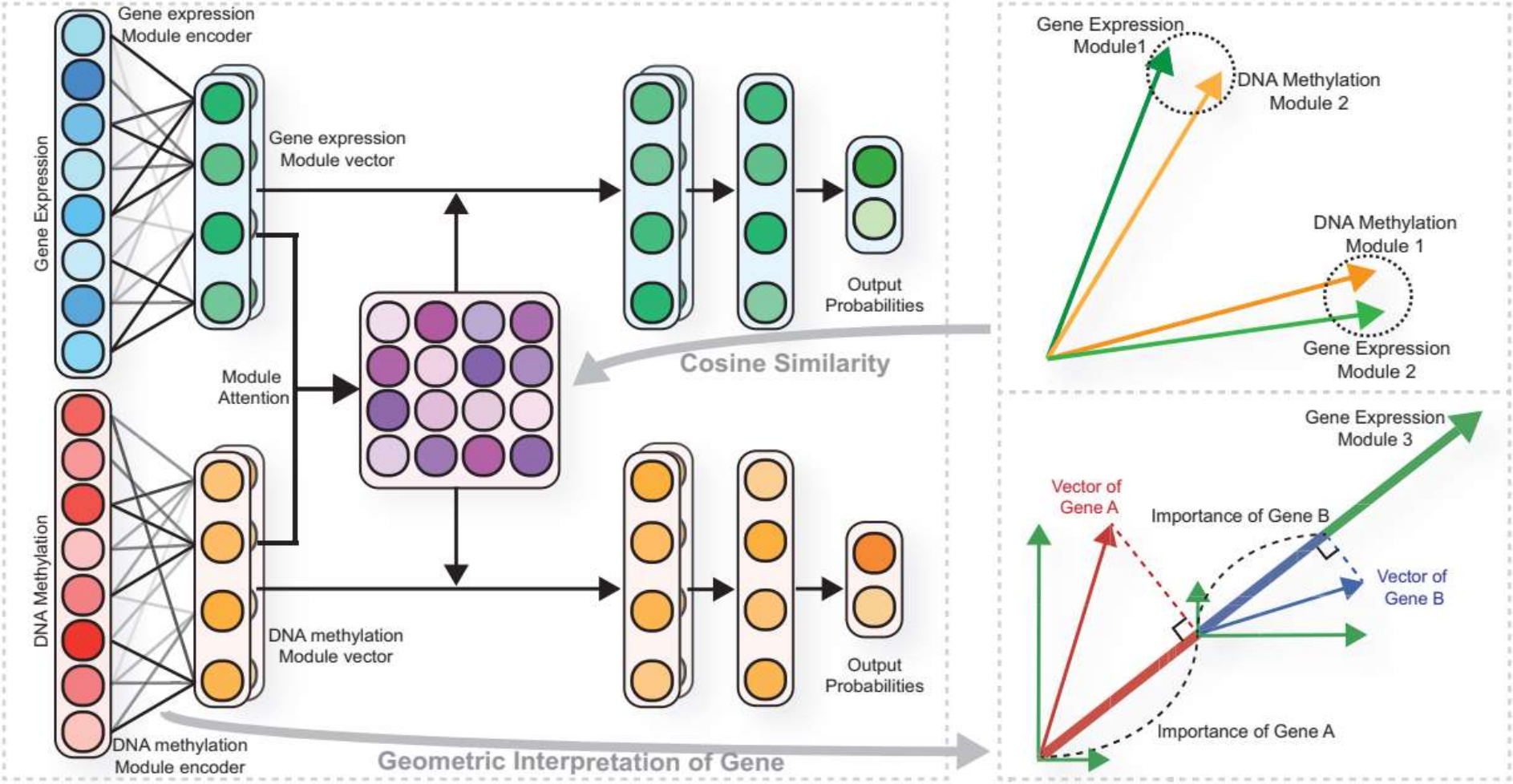
Generative Pretrained Transformers (GPT)





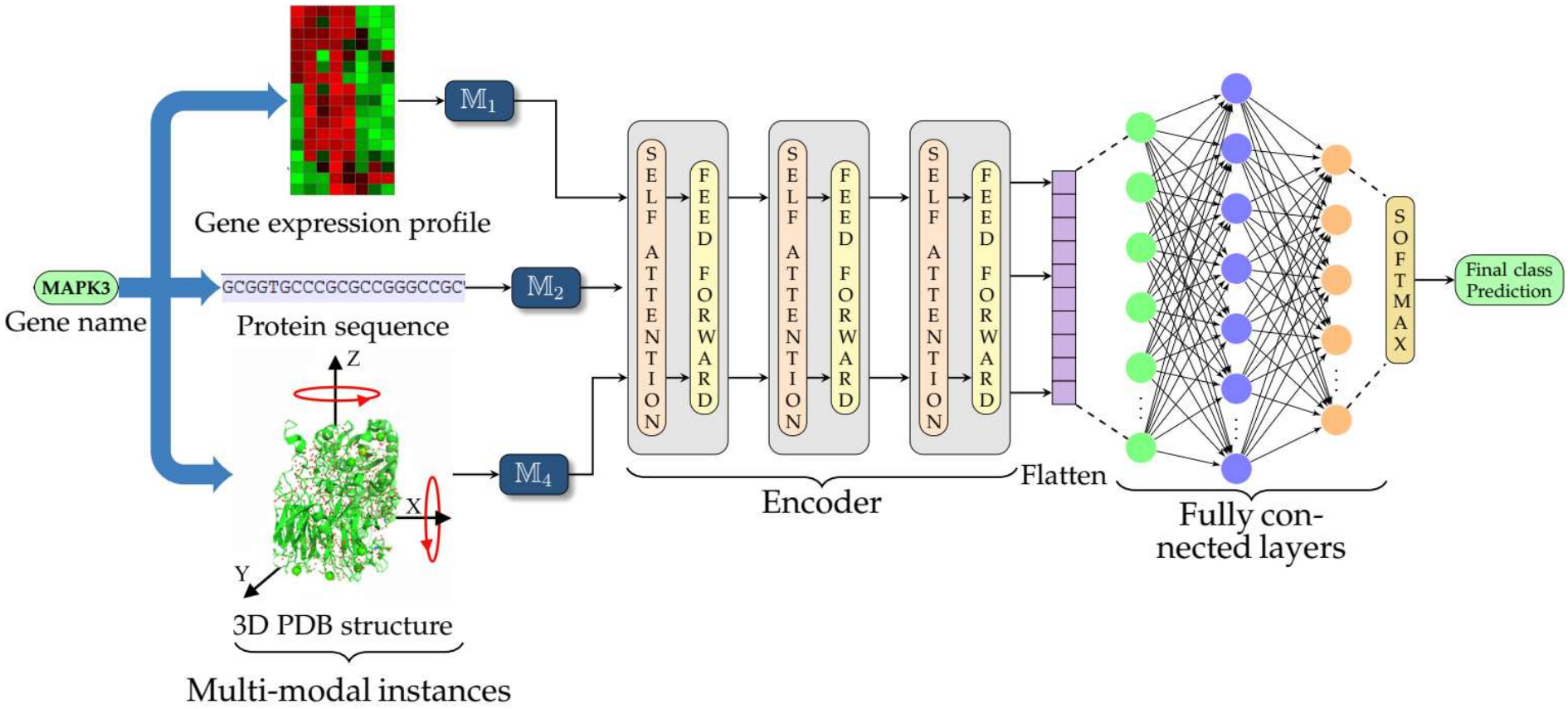
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Attention pour l'apprentissage multi-modal

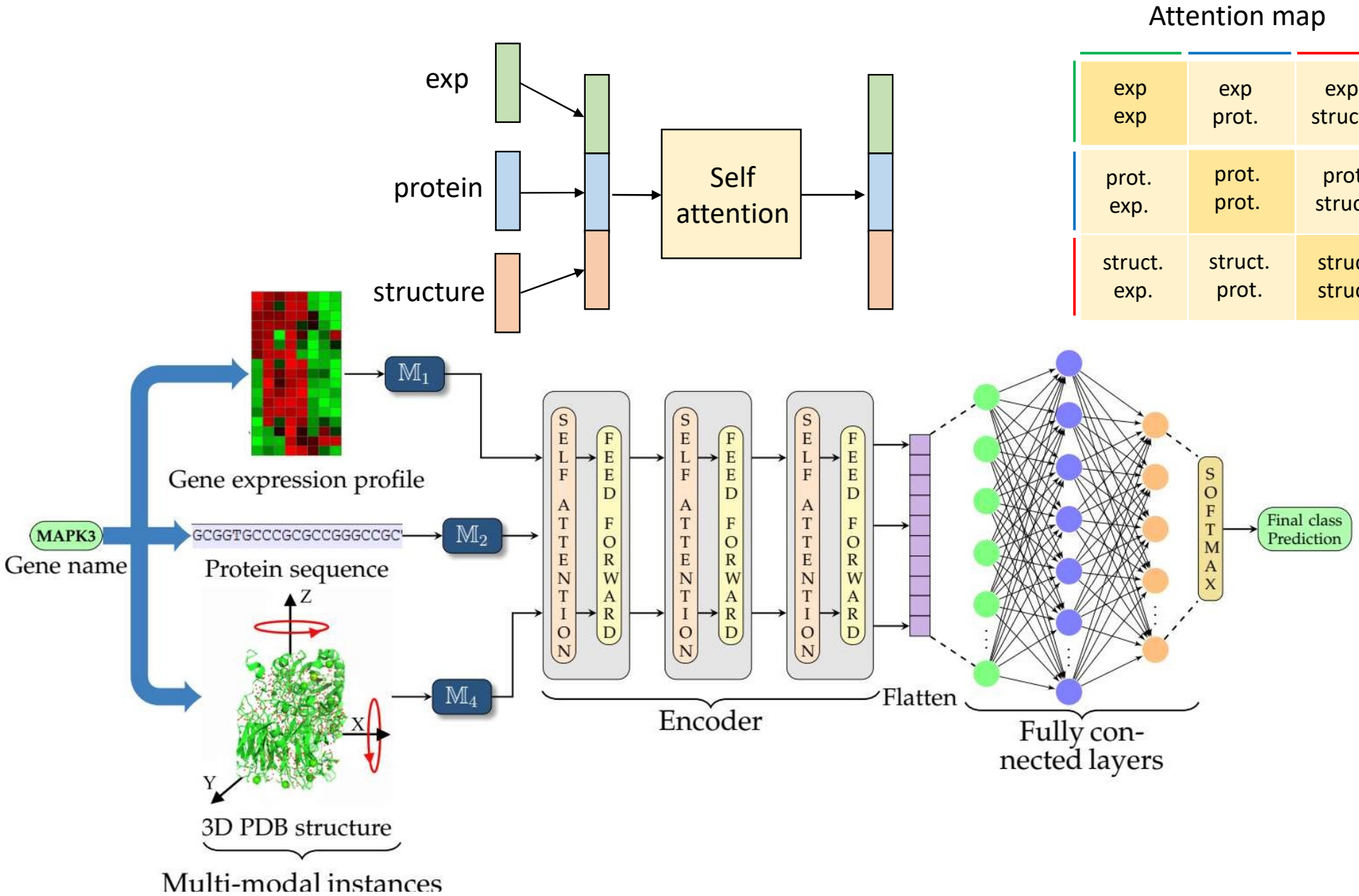


Moon, S., & Lee, H. (2022). MOMA: A Multi-Task Attention Learning Algorithm for Multi-Omics Data Interpretation and Classification. *Bioinformatics*.

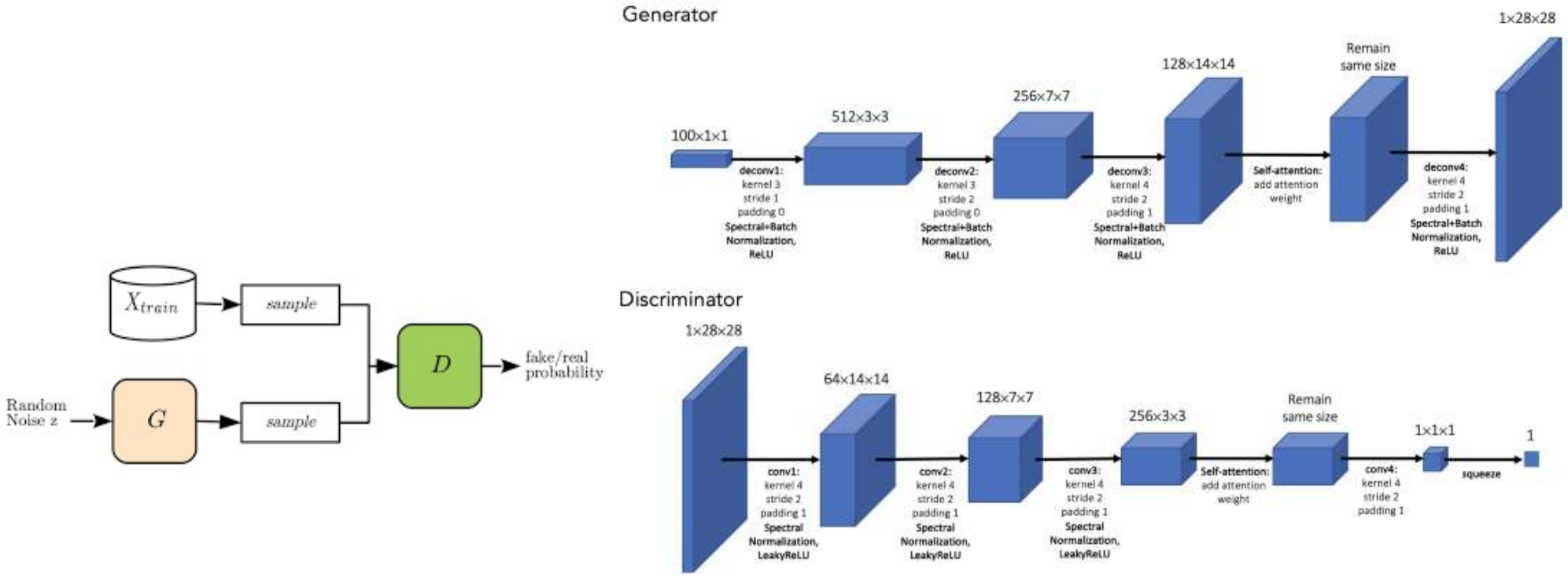
Self-attention pour l'apprentissage multi-modal



Dutta, P., Patra, A. P., & Saha, S. (2021). DeePROG: Deep Attention-based Model for Diseased Gene Prognosis by Fusing Multi-omics Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

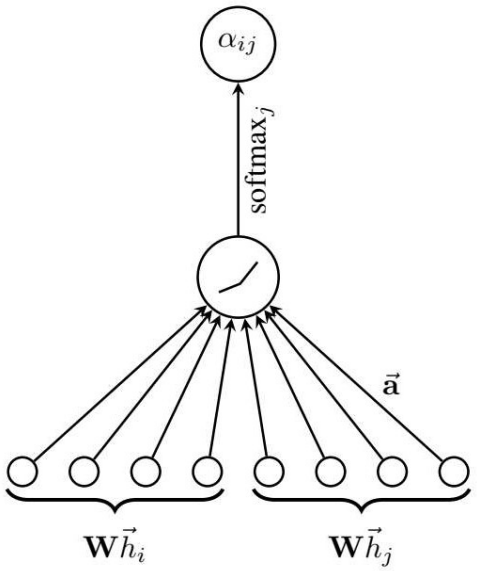

exp
protein
structure
Self attention
Attention map
exp exp
exp prot.
exp struct.
prot. exp.
prot. prot.
prot. struct.
struct. exp.
struct. prot.
struct. struct.
Gene expression profile
MAPK3
Gene name
GCGGTGCCCGCGCCGGGCCGC
Protein sequence
Z
X
Y
3D PDB structure
Multi-modal instances
M1
M2
M4
SELF ATTENTION
FEED FORWARD
Encoder
Flatten
SOFTMAX
Final class Prediction
Fully con-nected layers

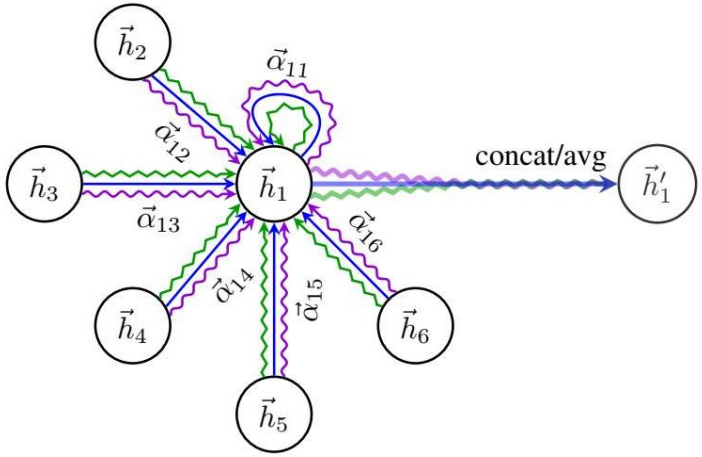
Self-Attention GAN (SAGAN)



Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, May). Self-attention generative adversarial networks In *International conference on machine learning* (pp. 7354-7363). PMLR

Graph Attention Network (GAT)





Valeurs d'attention

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Sortie (node embeddings)

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right).$$

Multi-heads attention

$$\vec{h}'_i = \Big\Vert_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Un problème de taille

	BERT	RoBERTa
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**