Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

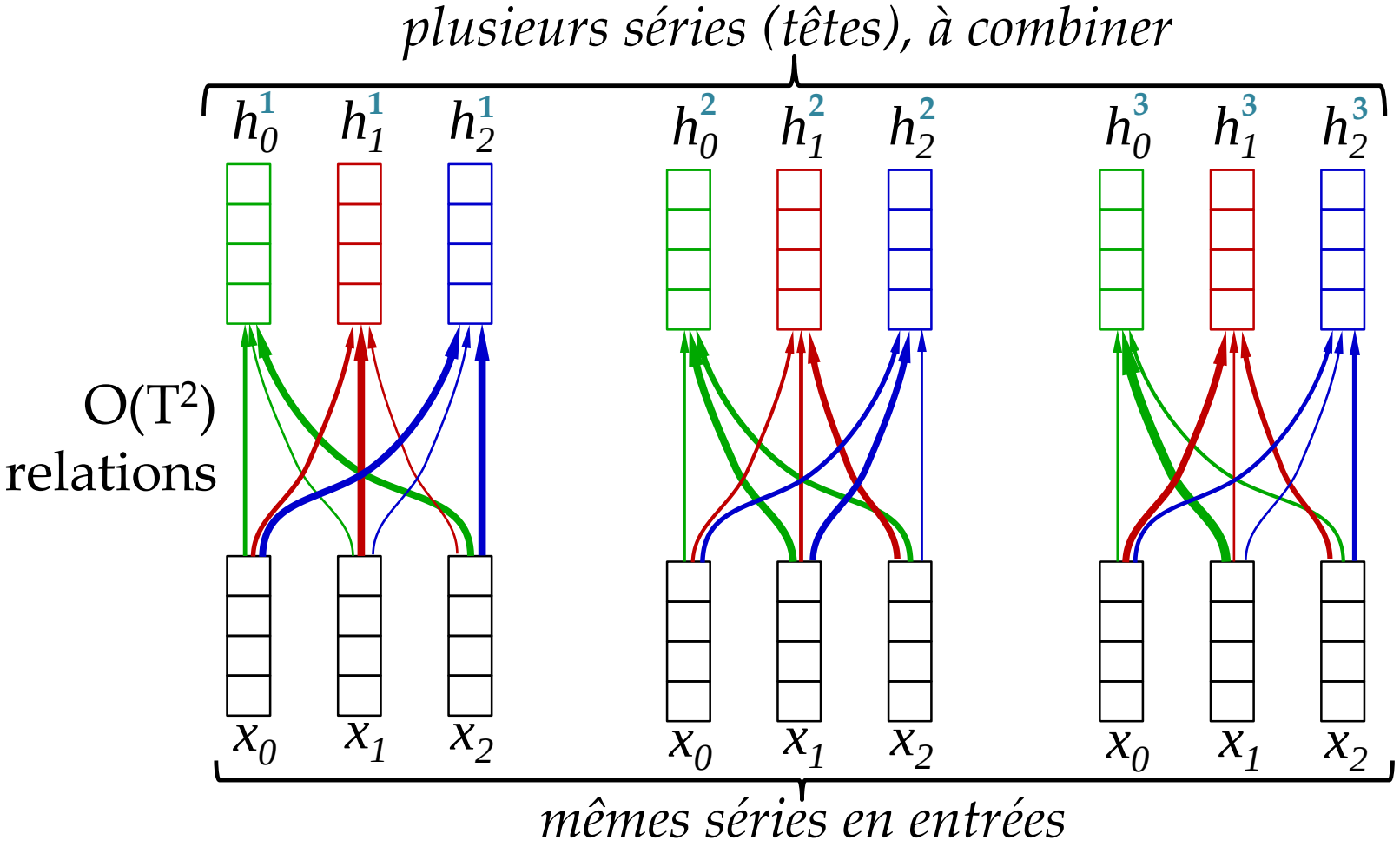
Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

NIPS 2017

Évolution

- *Word alignment* (précédent)
  - attention input-output
  - réseau *a*( ) calculant l'attention peu profond
- *Attention is all you need*
  - attention input-input, input-output, output-output
  - beaucoup plus de profondeur
  - Aucune récurrence
  - Plus facile à entraîner
    - Gradient se propage bien
    - Facilité à paralléliser (car non-séquentiel)
  - Utilise le self-attention

Multi-head : attentions combinées

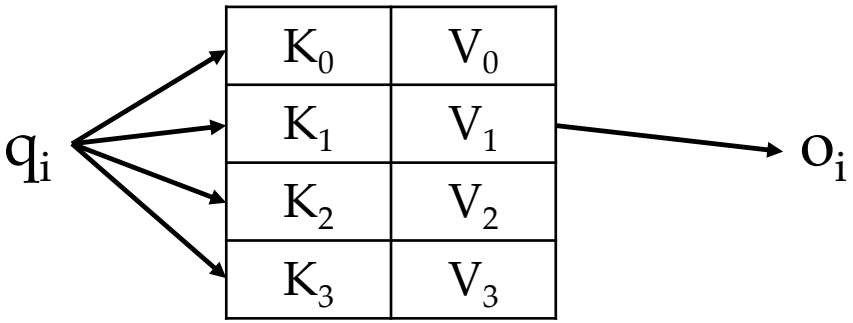


- Chaque tête peut apprendre des relations temporelles différentes (+ grande flexibilité)
- Interprétabilité des résultats

Single-head : mémoire associative

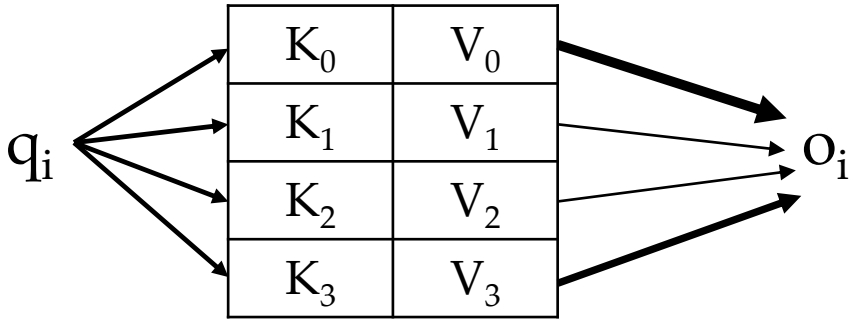
- Voir comme une mémoire associative, version *soft* d'un dictionnaire Python
  - clefs + valeurs
  - requête
  - fonction de distance requête-clefs

Python : hard



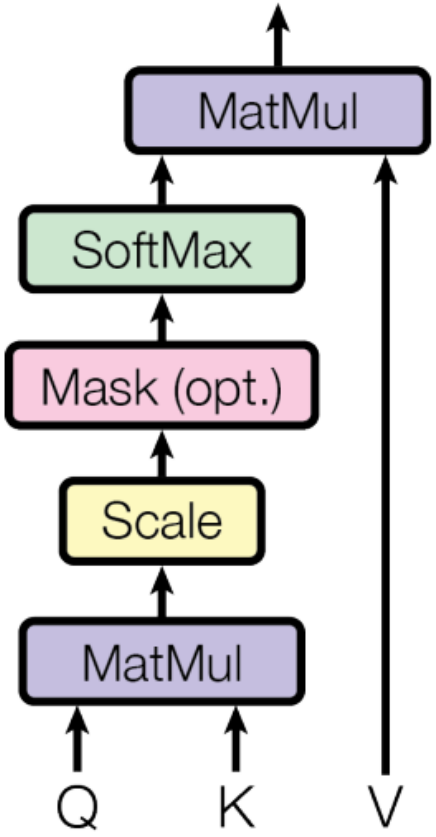
si $q_i=K_1$

soft



Single-head : mémoire associative

- Similarité cosinus (*cosine distance*)
- Utilisation du softmax pour les pondérations
- Compacter toutes les requêtes q_i dans une matrice Q
  - optimisation GPU pour matrice-matrice malgré le facteur $O(T^2)$



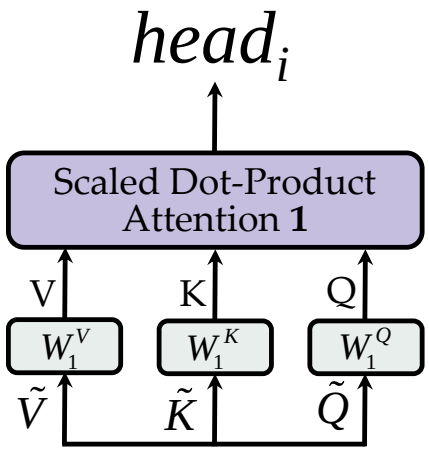
$$Attention(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

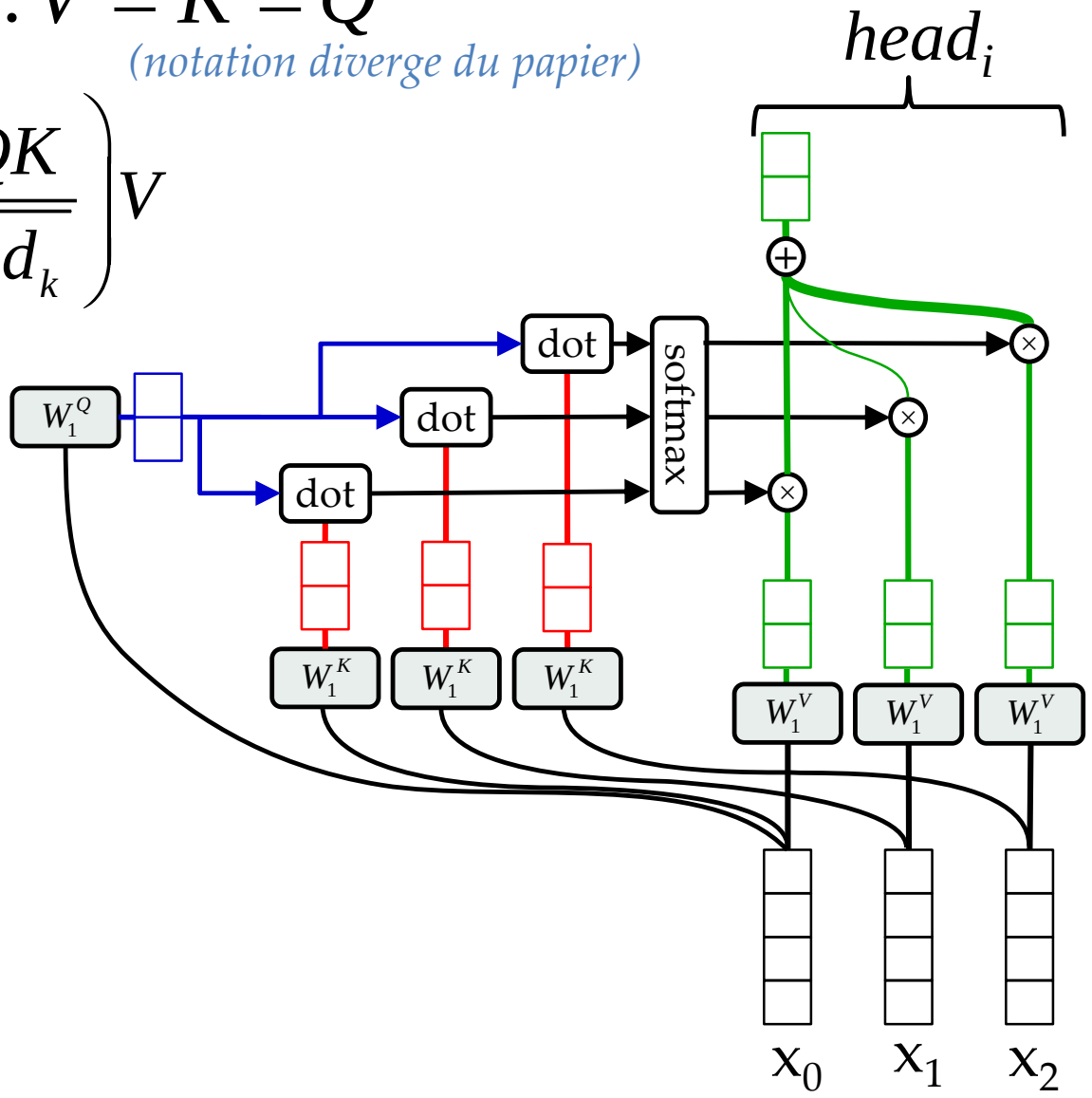
$head_i$: Scaled Dot-Product Attention

- Self-attention : $\tilde{V} = \tilde{K} = \tilde{Q}$

(notation diverge du papier)

$$sortie = \text{Softmax}\left(\frac{QK}{\sqrt{d_k}}\right)V$$



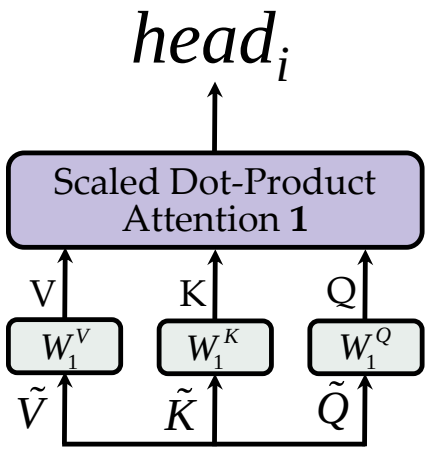


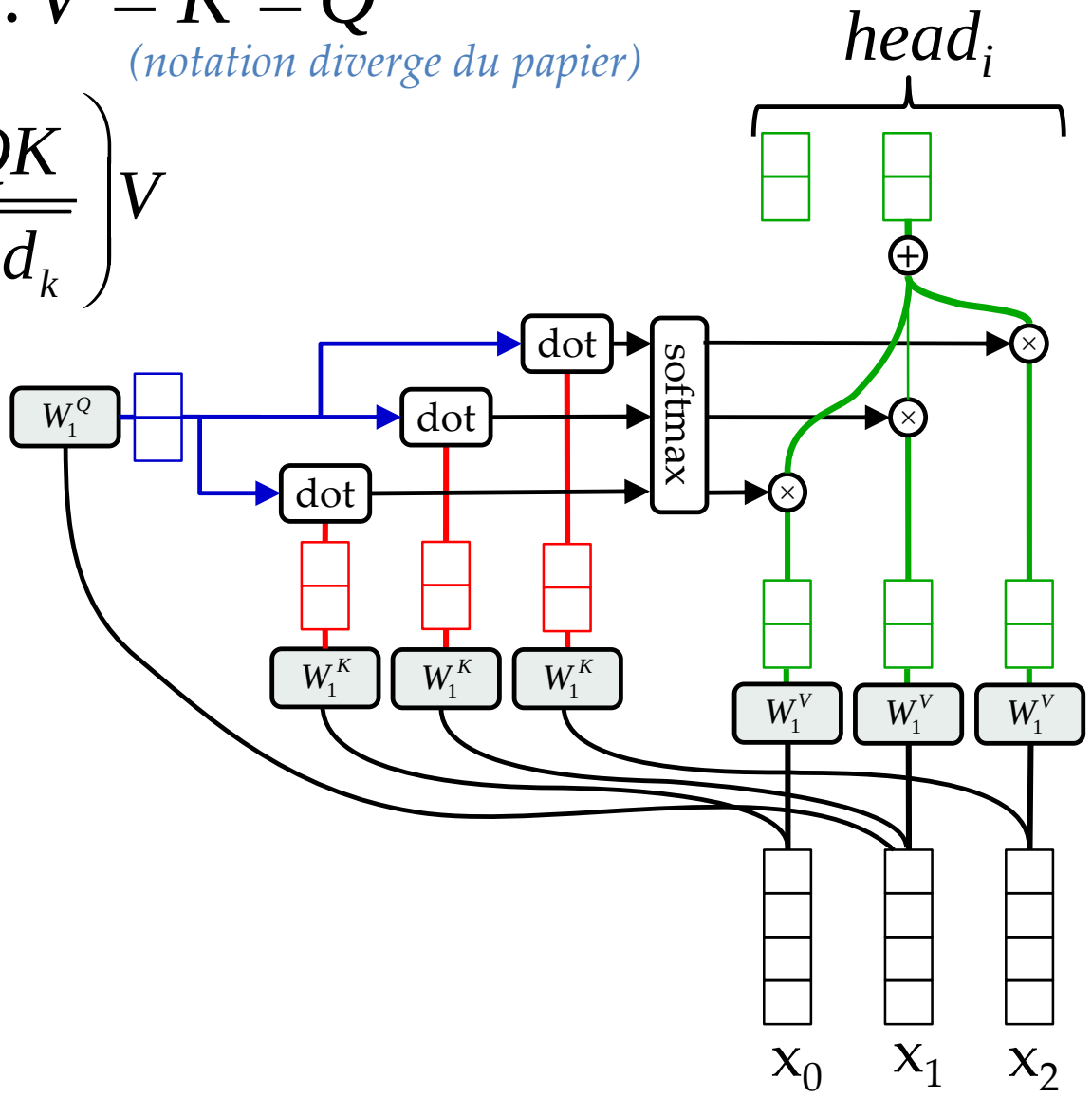
$head_i$: Scaled Dot-Product Attention

- Self-attention : $\tilde{V} = \tilde{K} = \tilde{Q}$

(notation diverge du papier)

$$sortie = \text{Softmax}\left(\frac{QK}{\sqrt{d_k}}\right)V$$



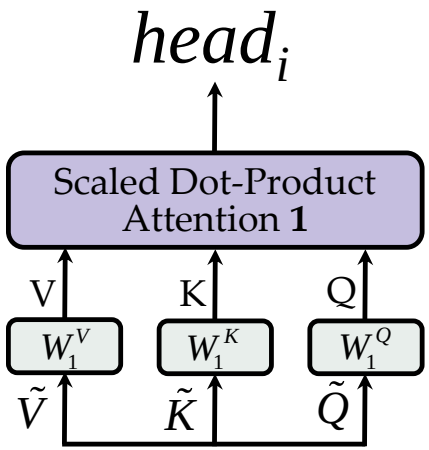


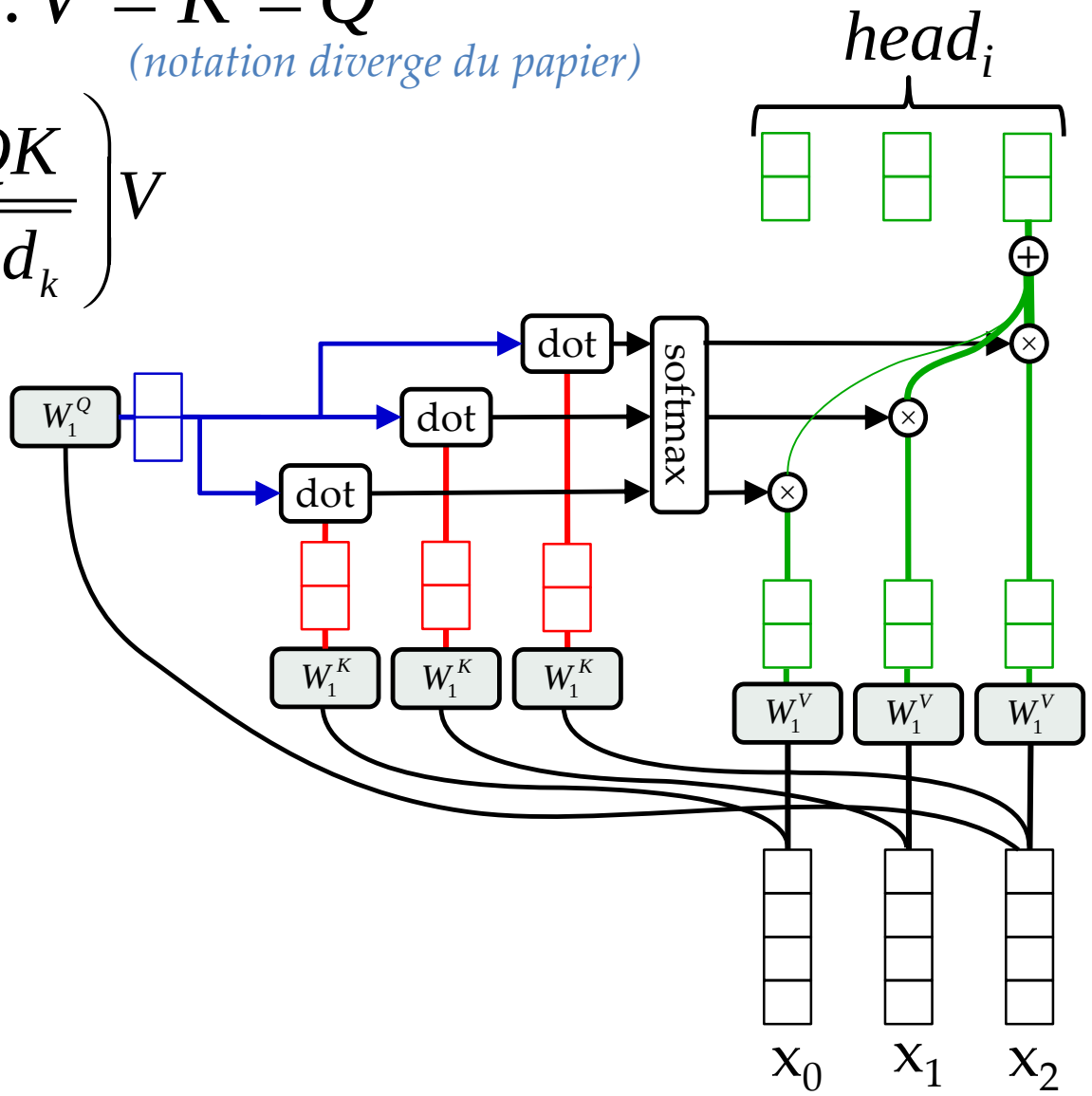
$head_i$: Scaled Dot-Product Attention

- Self-attention : $\tilde{V} = \tilde{K} = \tilde{Q}$

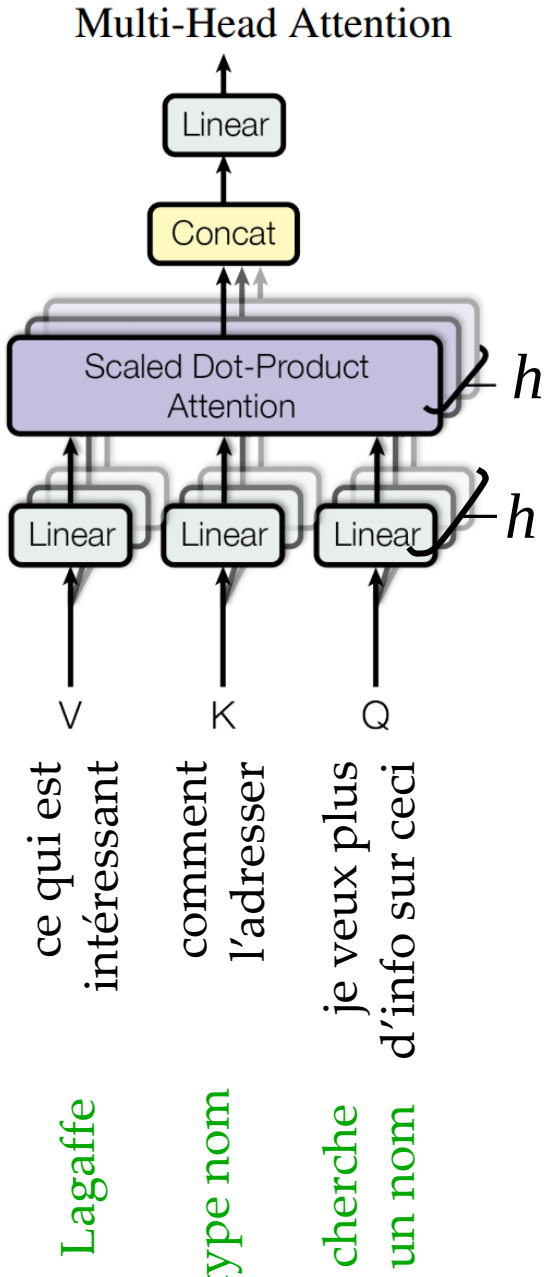
(notation diverge du papier)

$$sortie = \text{Softmax}\left(\frac{QK}{\sqrt{d_k}}\right)V$$





Attention


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
h
V
K
Q
ce qui est intéressant
comment l'adresser
je veux plus d'info sur ceci
Lagaffe
type nom
cherche un nom

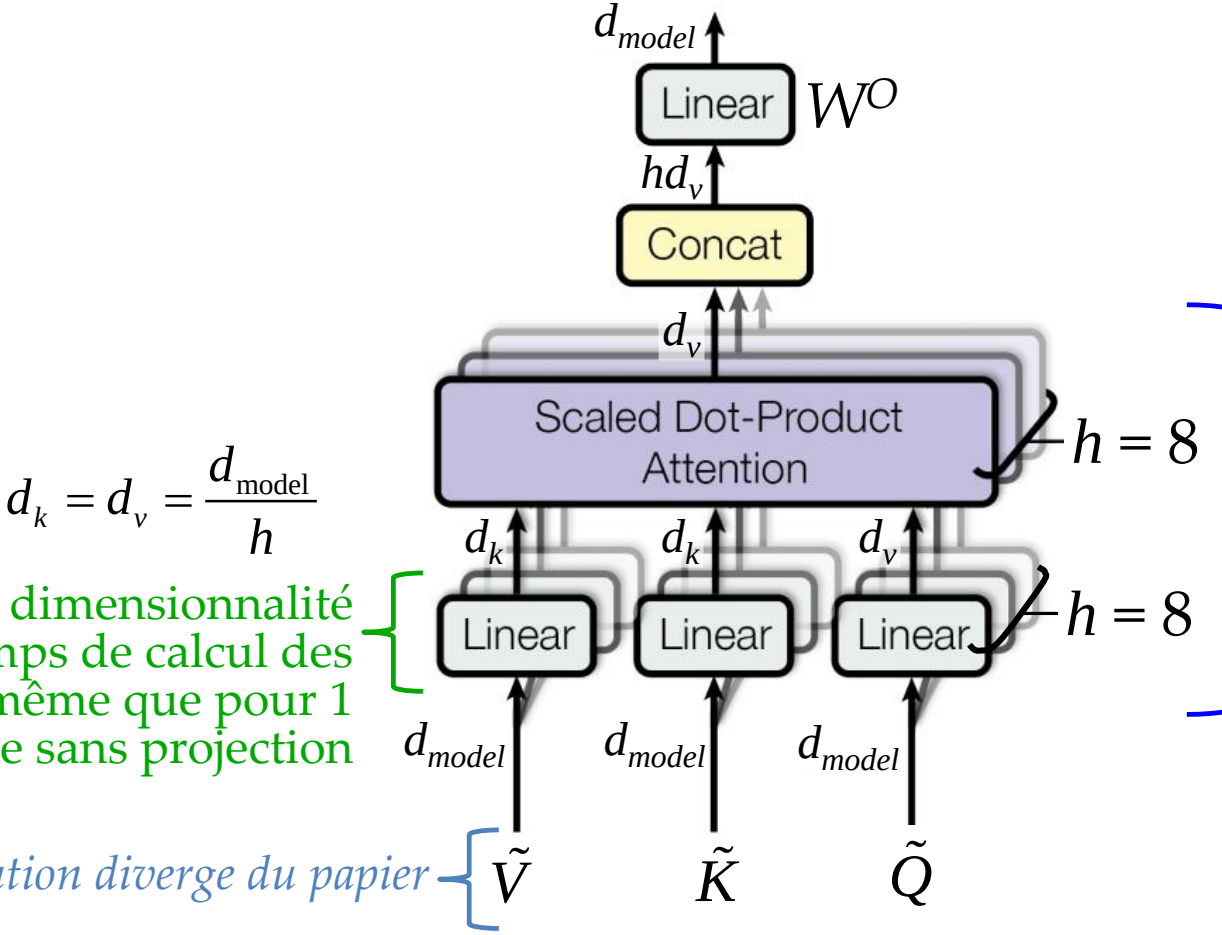
Avantages du self-attention

- Facilité à paralléliser le calcul
  - RNN est fondamentalement séquentiel
- Longueur **fixe** du chemin dans le graphe de calcul pour les dépendances à longue-portée
  - plus de vanishing gradient
  - RNN : longueur dépend du nombre d'itérations

Multi-head attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(\tilde{Q}W_i^Q, \tilde{K}W_i^K, \tilde{V}W_i^V)$$

notation diverge du papier



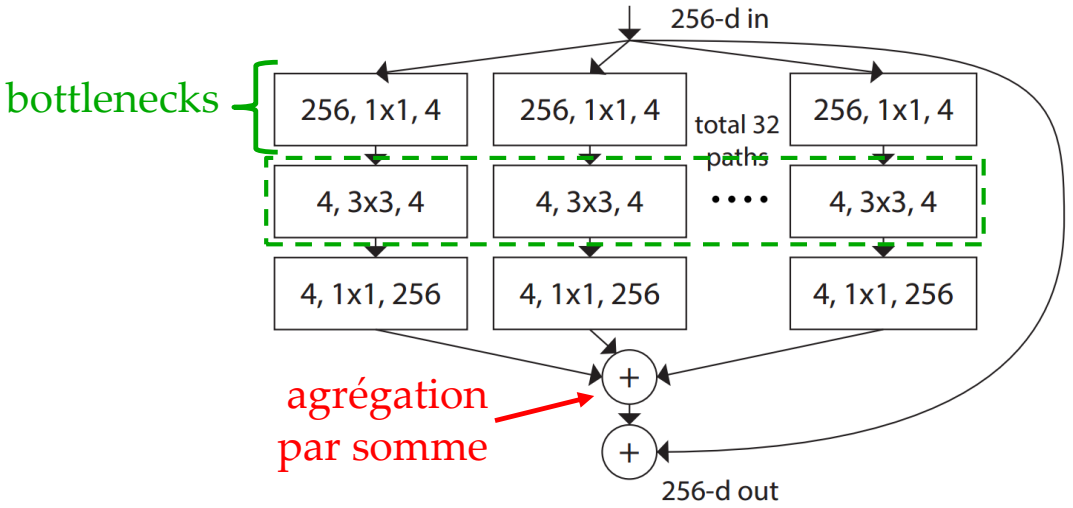
$$d_k = d_v = \frac{d_{\text{model}}}{h}$$

Réduction de dimensionnalité linéaire : temps de calcul des têtes est le même que pour 1 tête sans projection

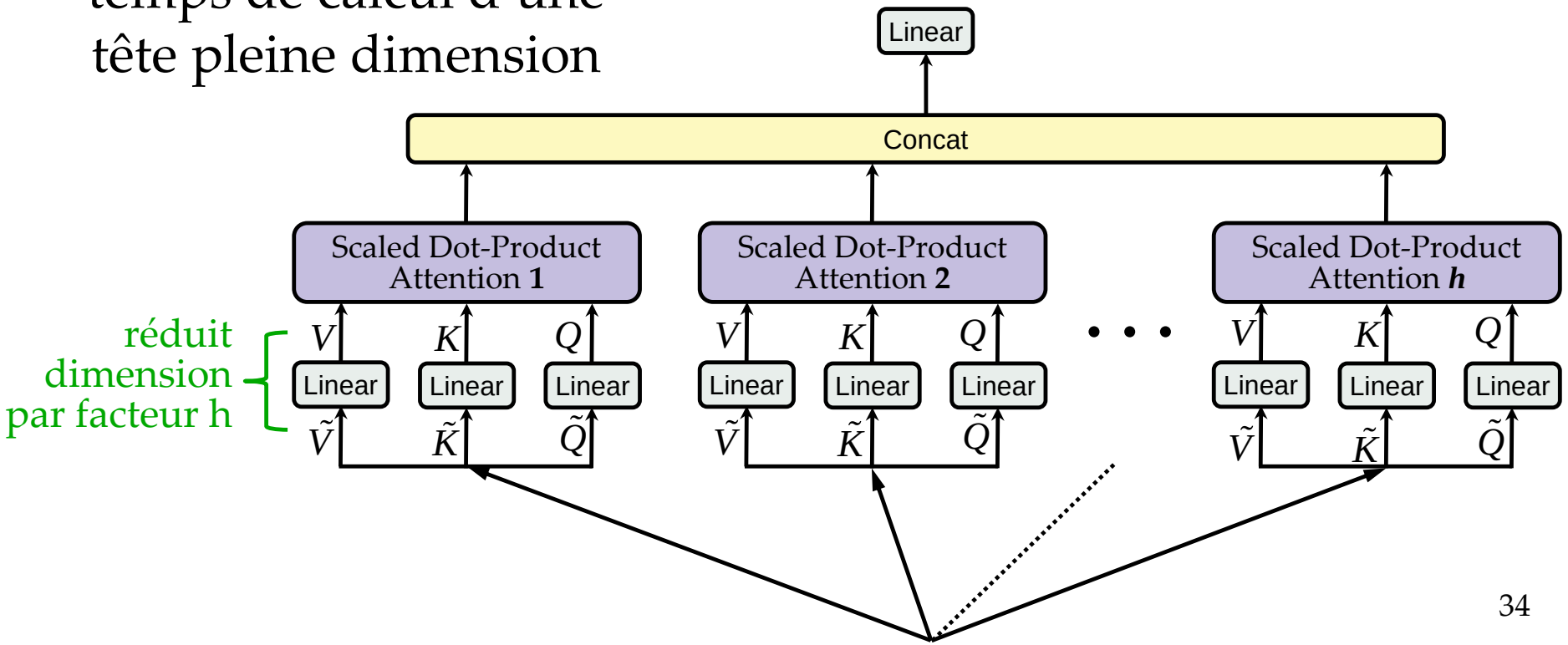
Multi-branche

notation diverge du papier

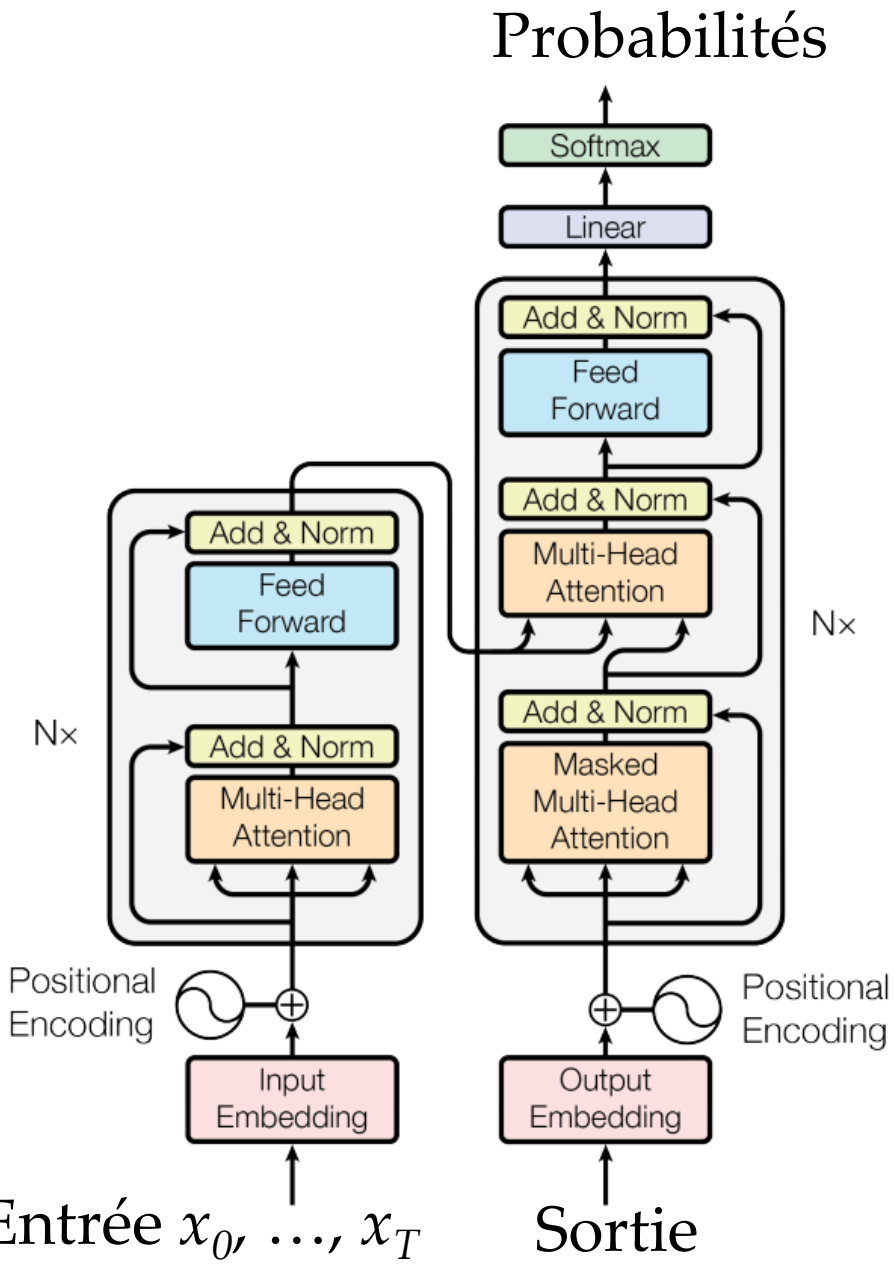
Similarité avec ResNext



Temps de calcul de h têtes sur dimension 1/h *égale* temps de calcul d'une tête pleine dimension

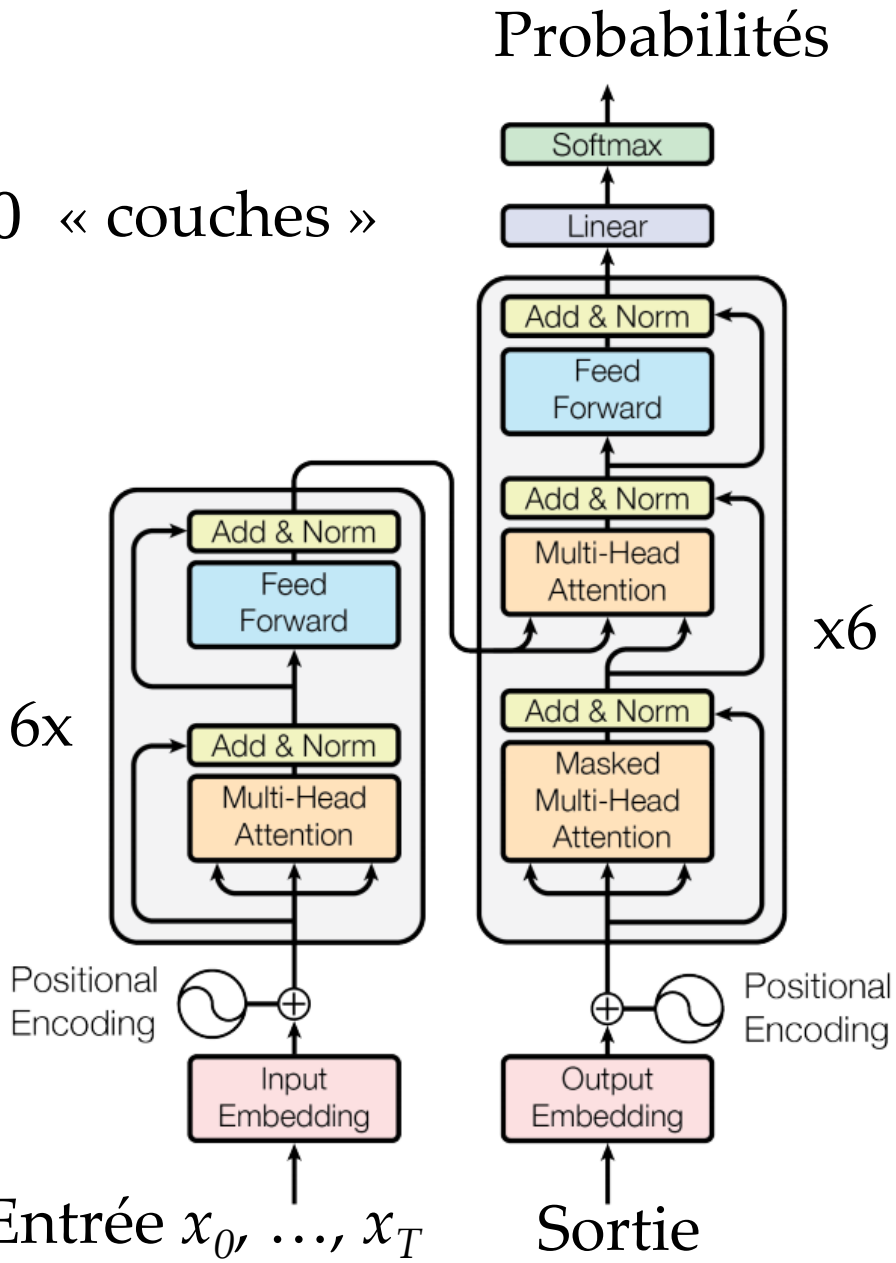


Architecture complète

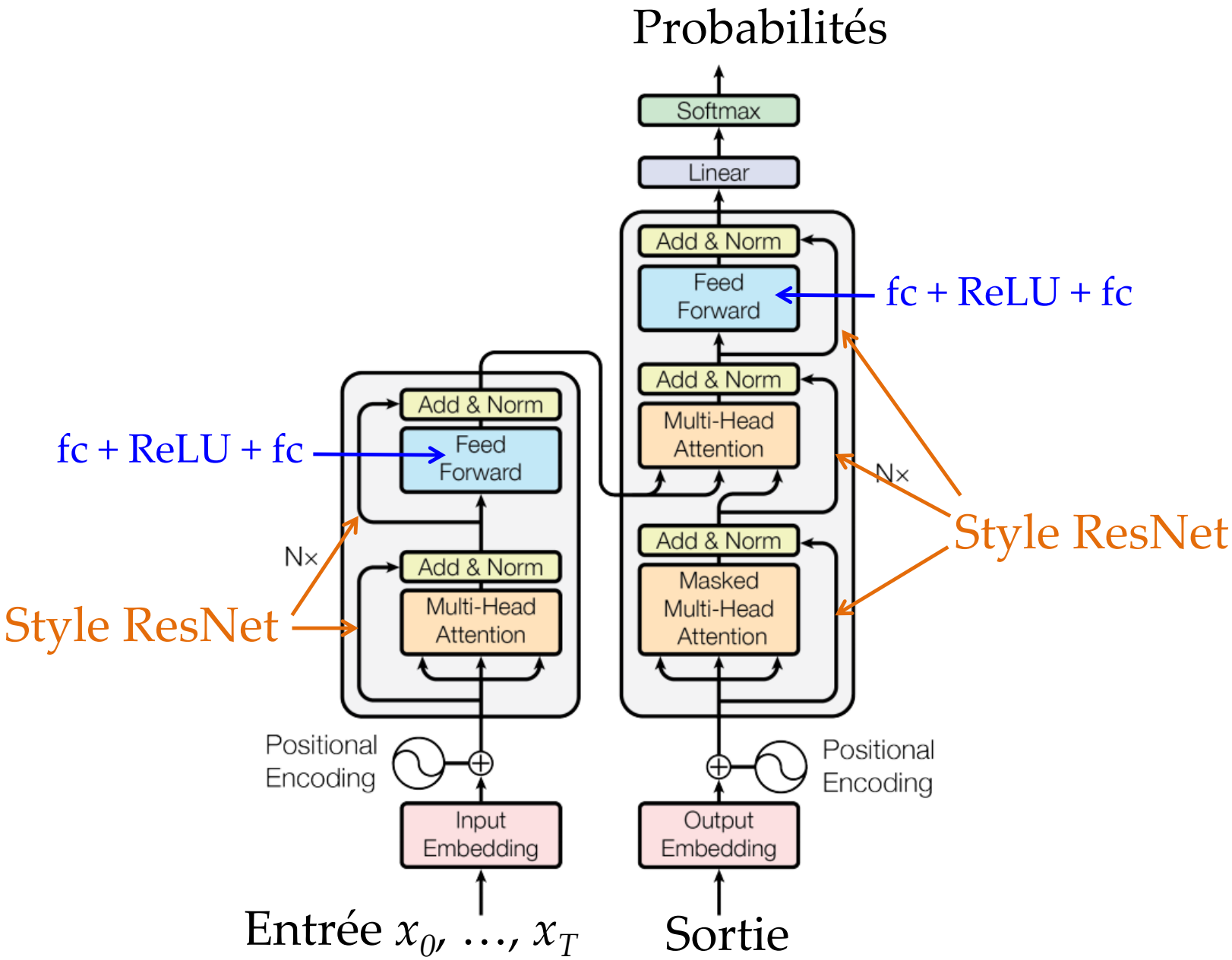


Profondeur

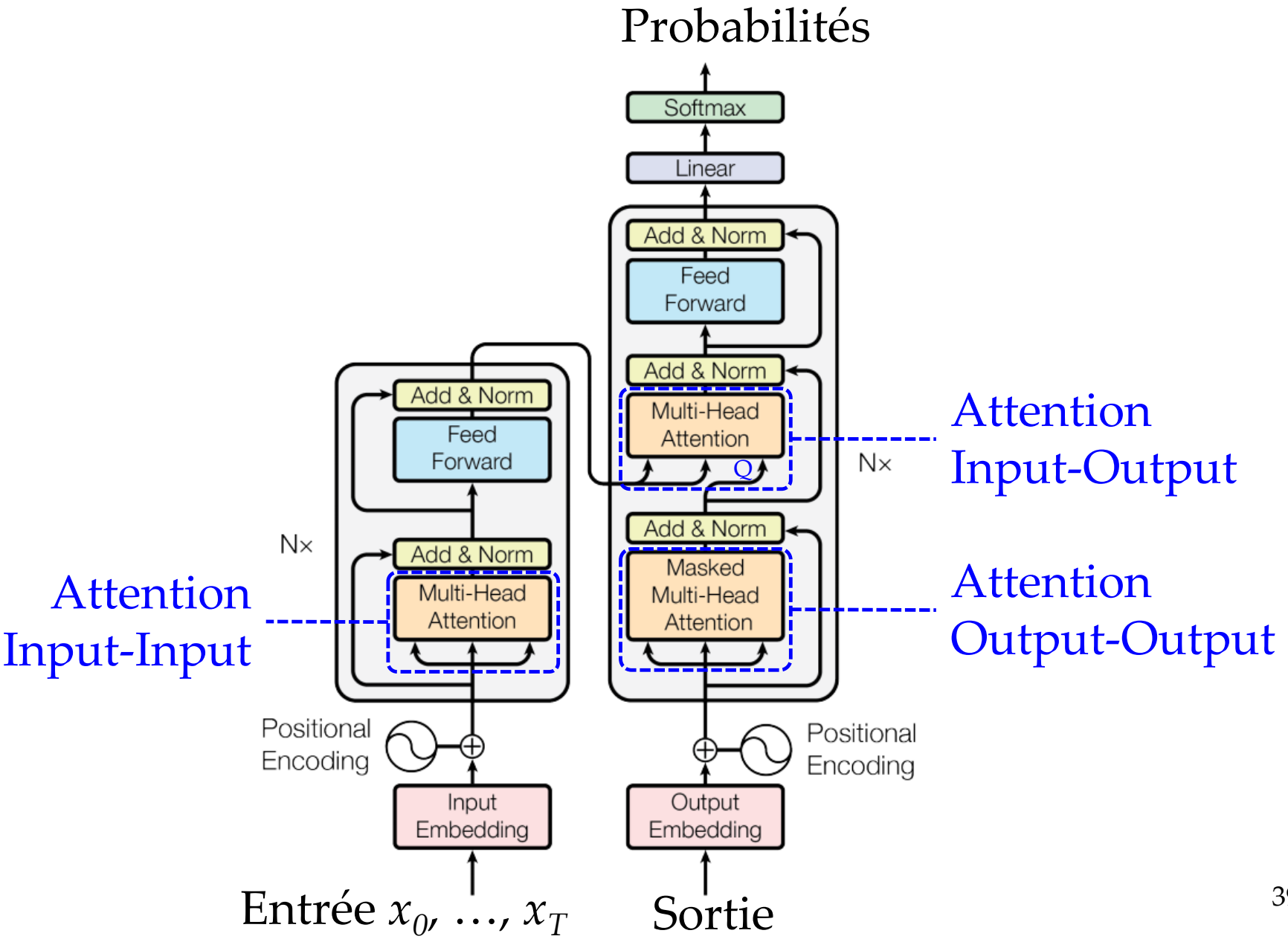
$6 \times 2 + 6 \times 3 = 30$ « couches »



Architecture

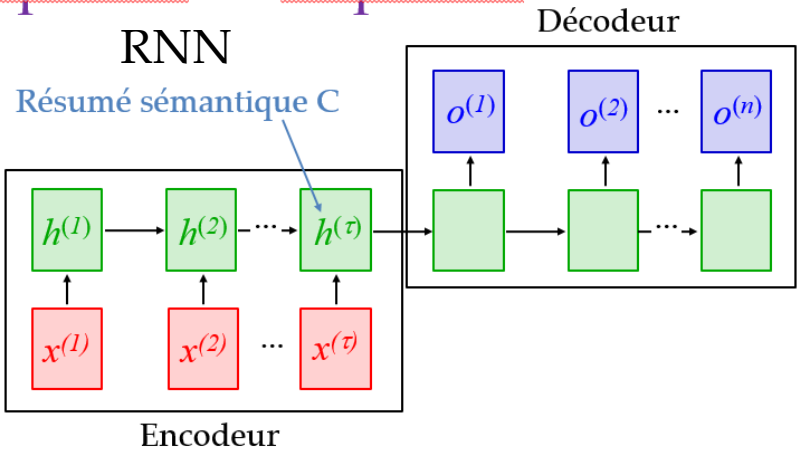


Répartition de l'attention

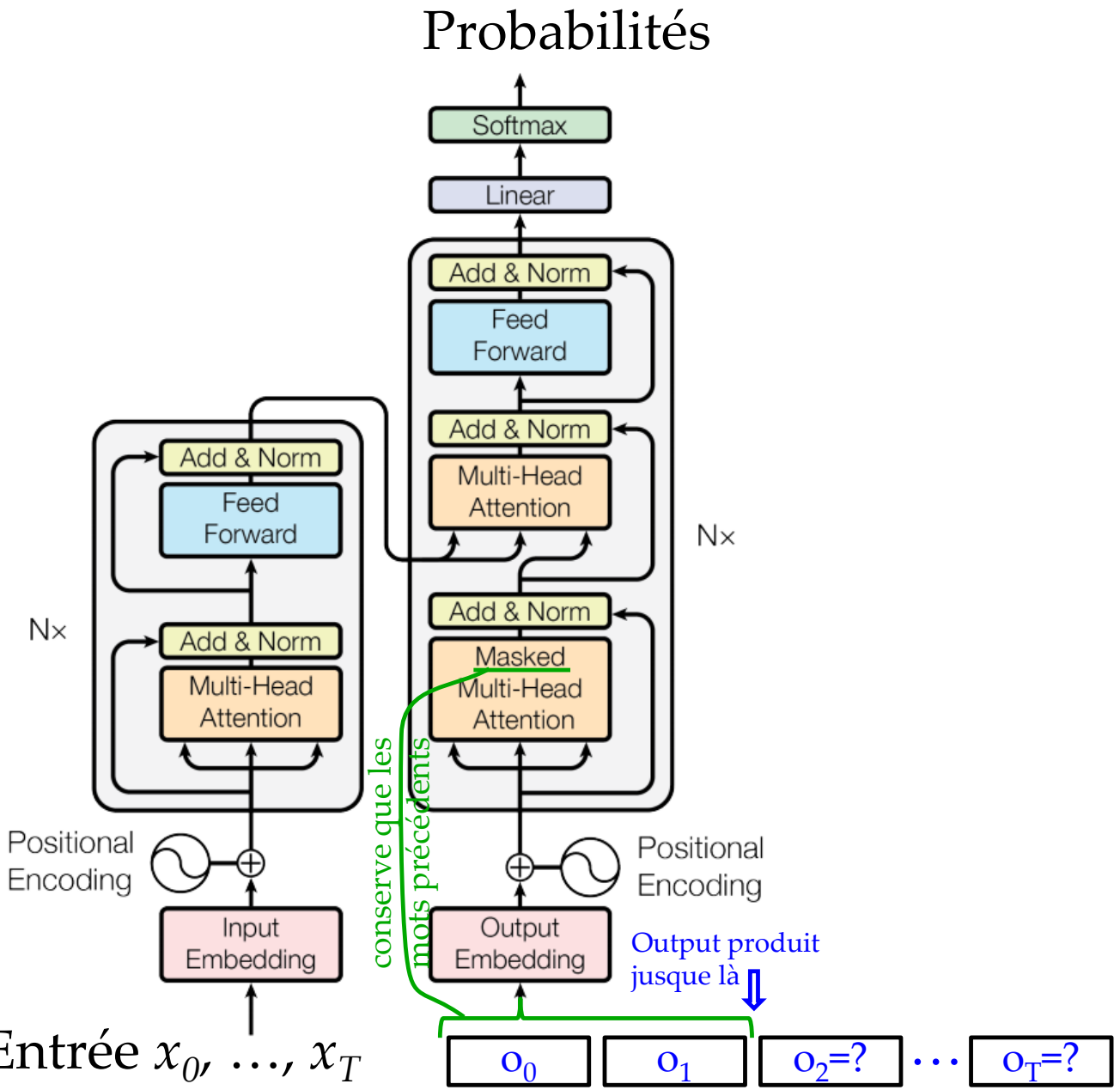


Vue encodeur-décodeur

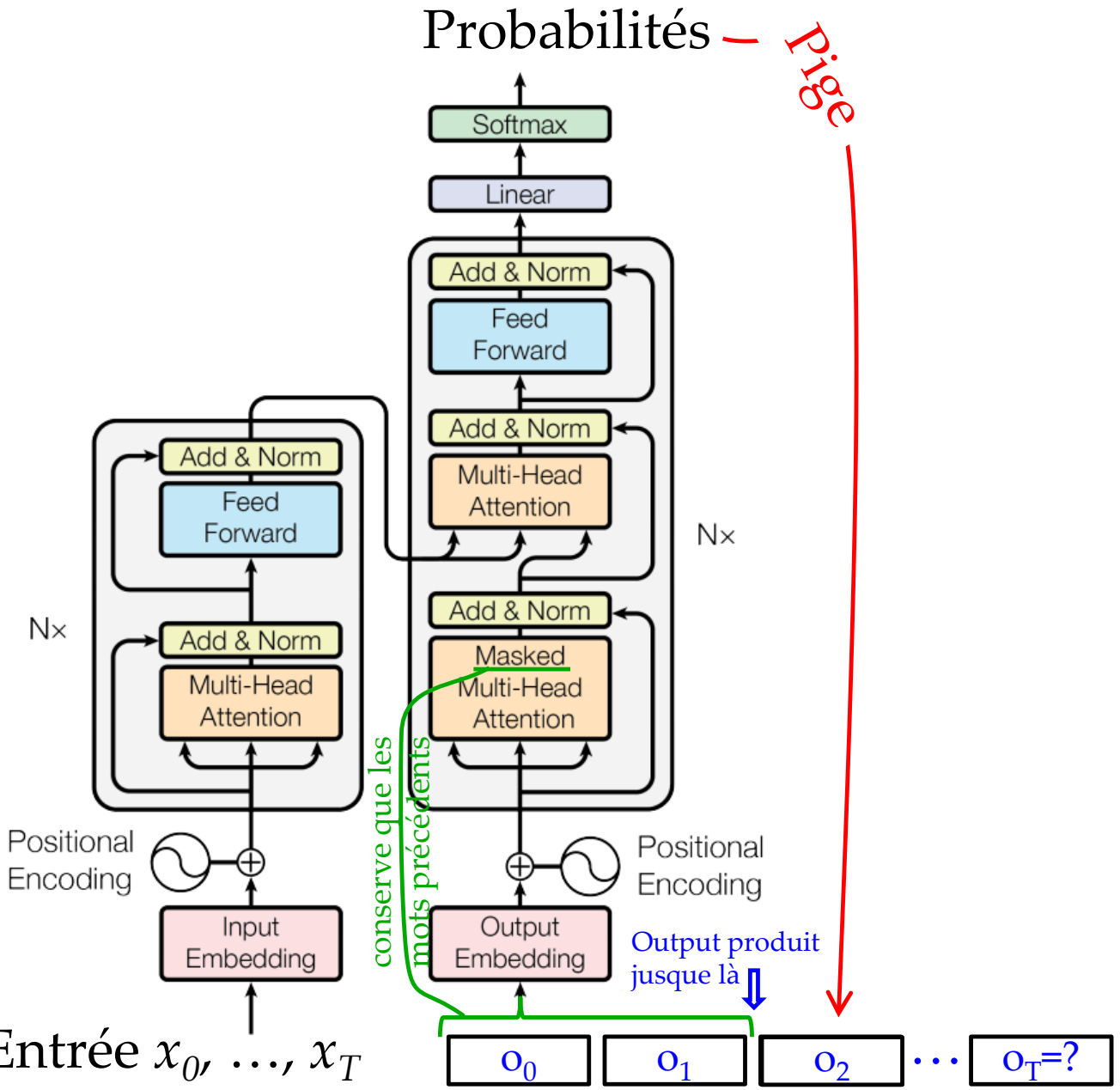
Sequence-to-sequence



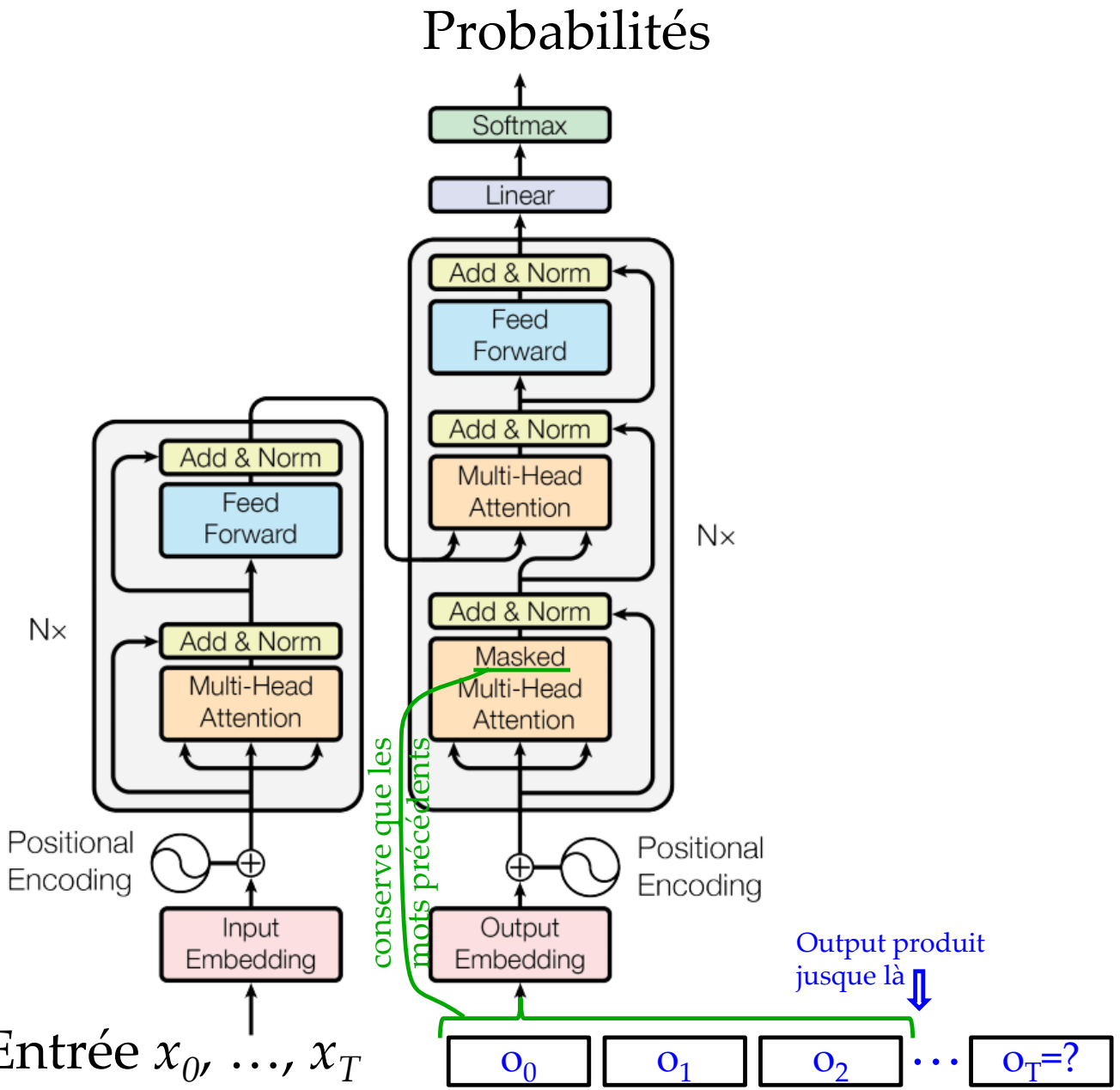
Génération séquence de sortie o



Génération séquence de sortie o

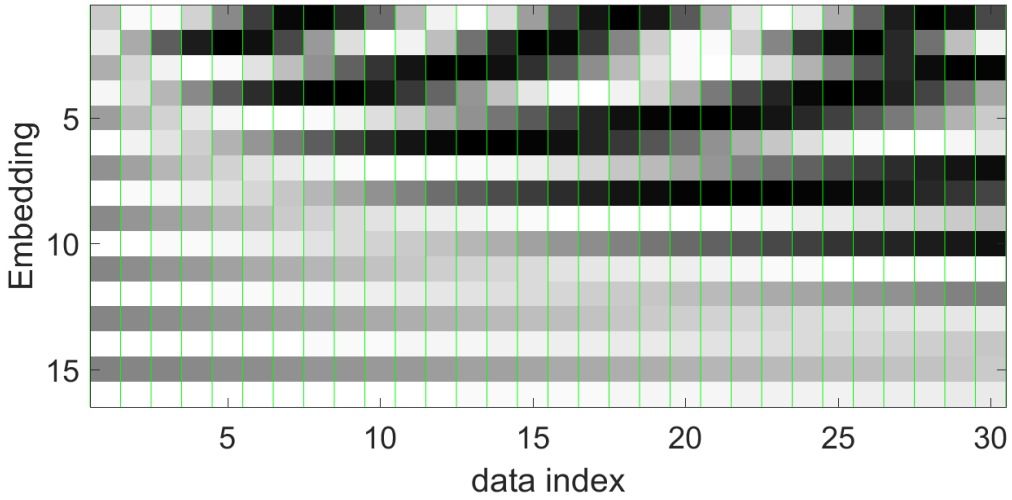


Génération séquence de sortie o



Encodage position sinus/cosinus

Code de position

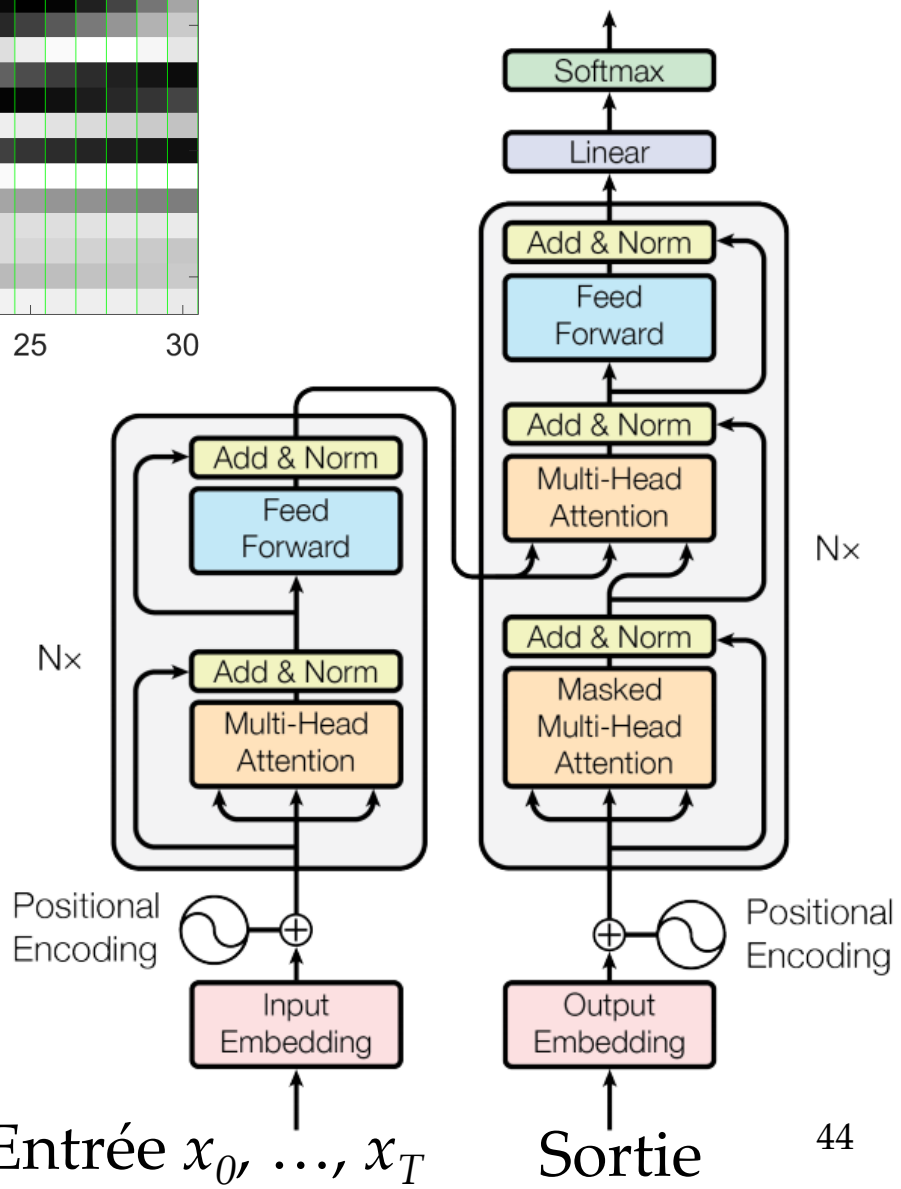


- Perte de l'ordre car l'approche est similaire à CBOW (continuous bag-of-words)
- Additionne à l'*embedding* un vecteur encodant les positions

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Redonne un signal sur l'ordre des mots



Résultats

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	