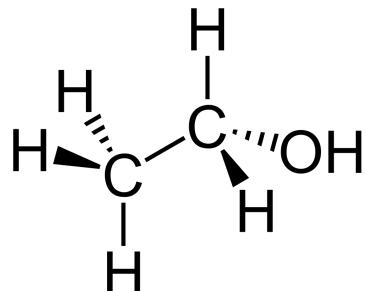

Machine Learning : Nouveaux réseaux profonds

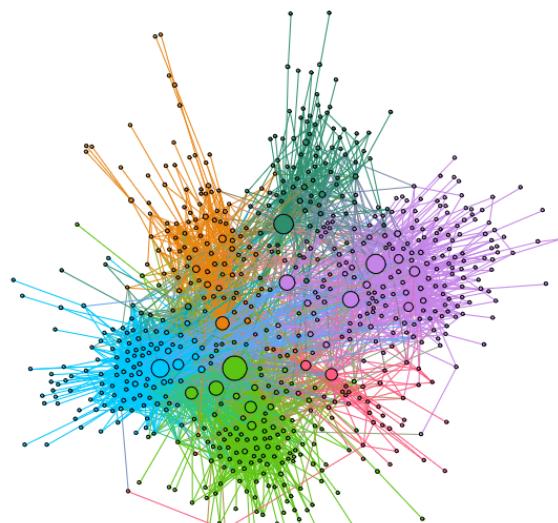
Graph Neural Network

Objectif : adapter la structure de notre réseau à la structure de nos données.

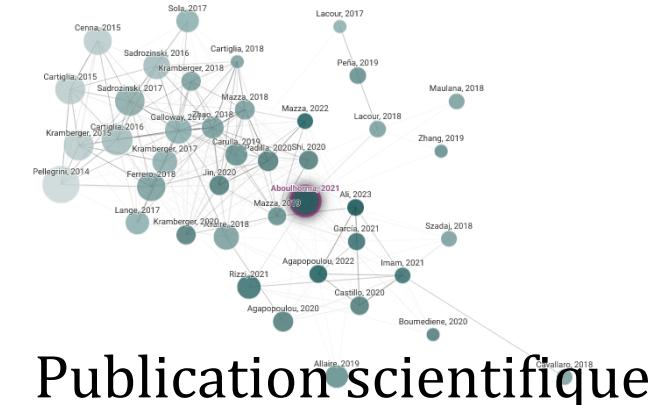
Grand nombre
de structure
représentable
par des graphes



Molécule



Email communication



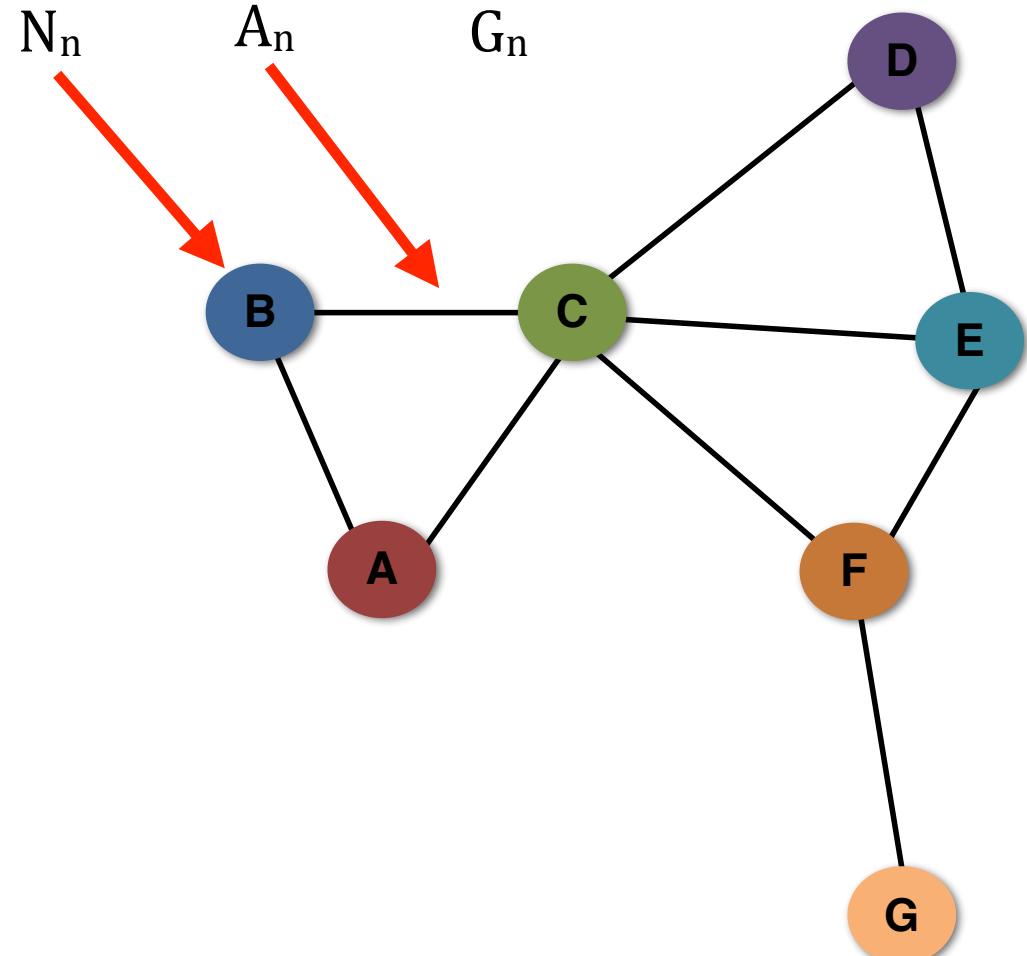
Publication scientifique



Relation sociale

Graph Neural Network

Graphe : 3 type d'info : Noeuds, arrêtes et globale

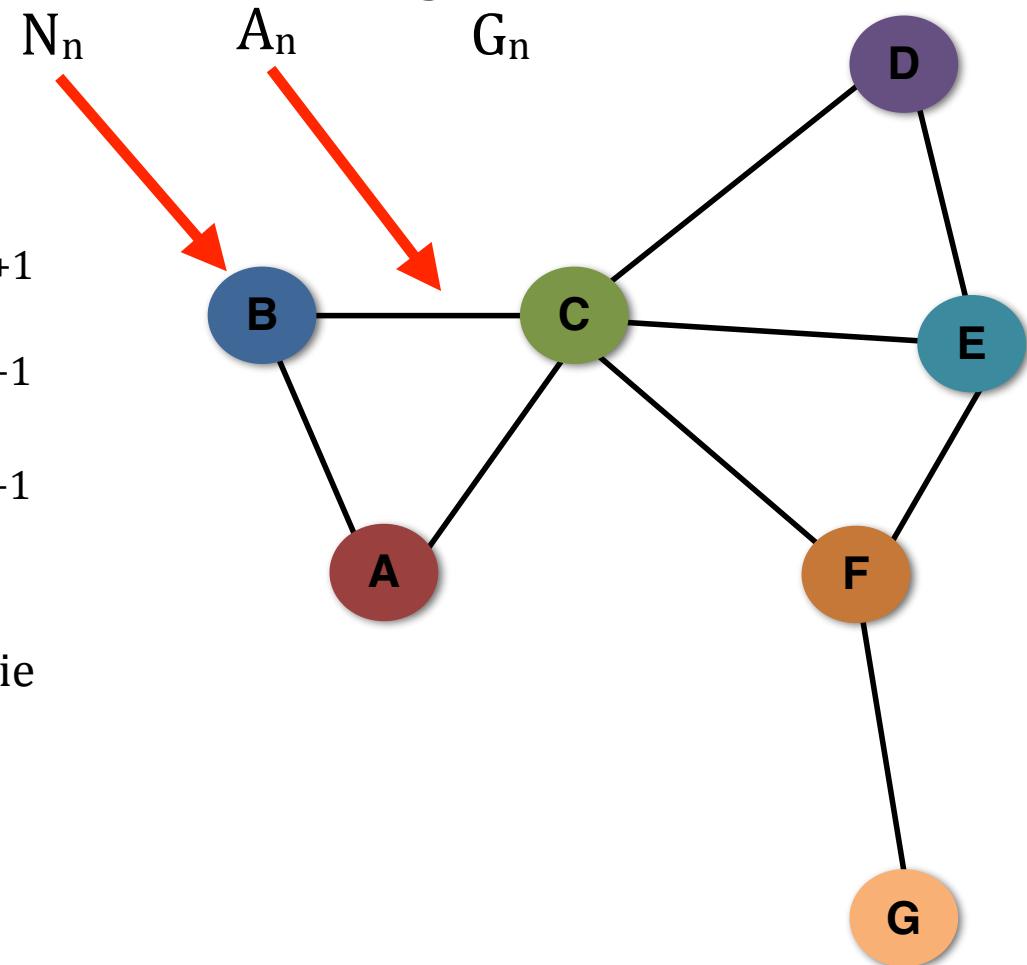
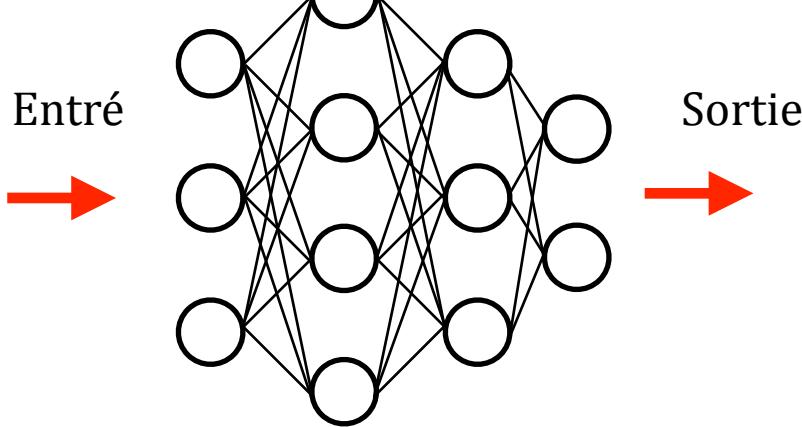


Graph Neural Network

Graphe : 3 type d'info : Noeuds, arrêtes et globale

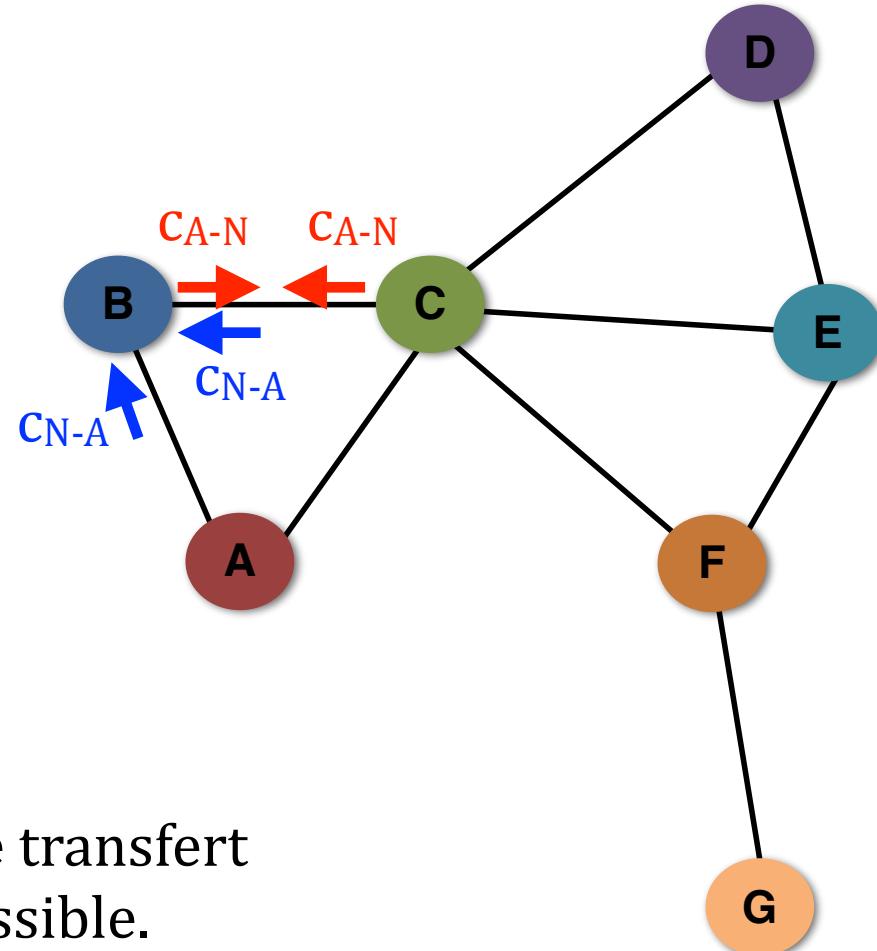
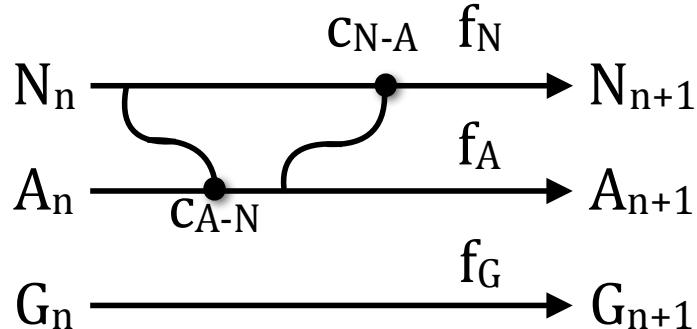
Application d'un NN à ces variables :

$$\begin{aligned} N_n &\xrightarrow{f_N} N_{n+1} \\ A_n &\xrightarrow{f_A} A_{n+1} \\ G_n &\xrightarrow{f_G} G_{n+1} \end{aligned}$$



Graph Neural Network : Message Passing

Transfert d'information entre les éléments du graph avec une fonction d'agrégation :



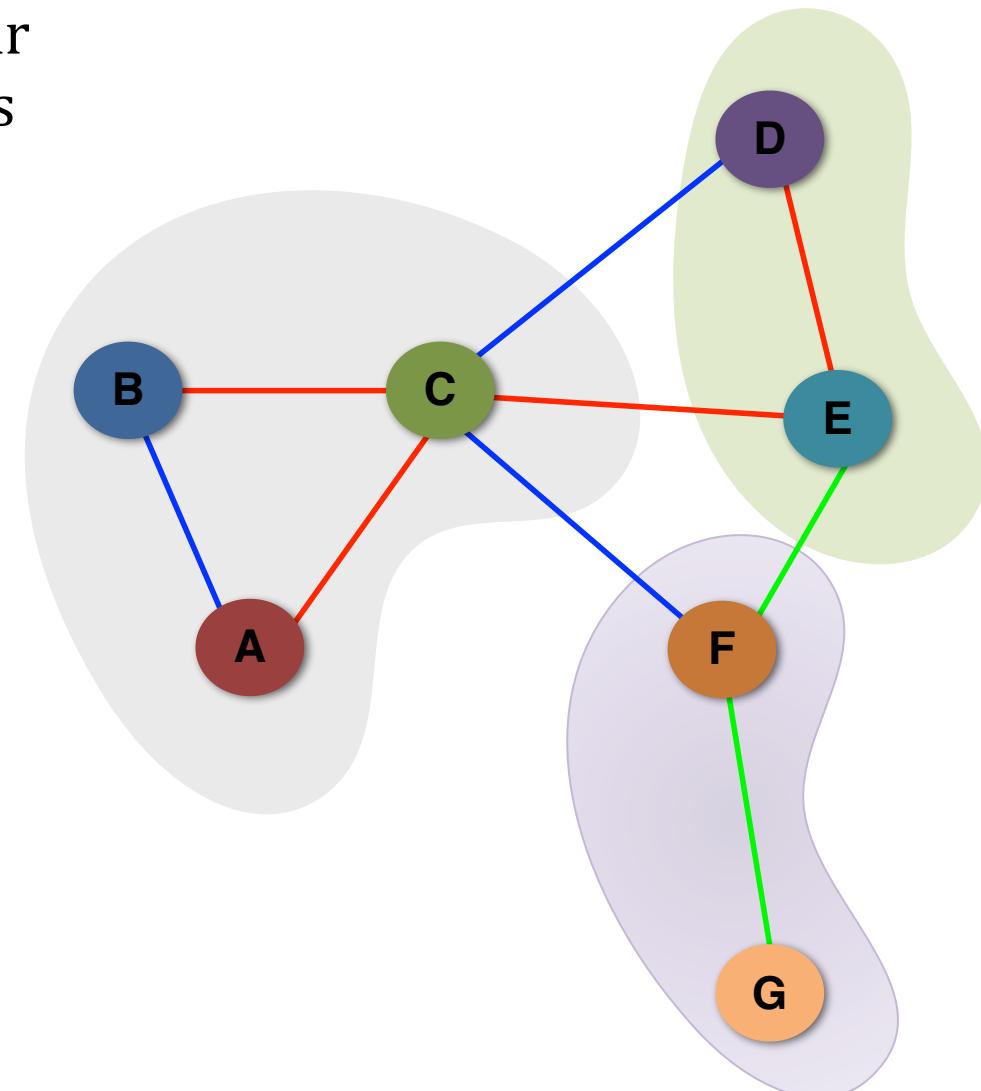
Différente méthode de transfert d'information sont possible.

Graph Neural Network : Classification

Utilisation d'un dernier NN pour la classification des Noeuds, des arrêtes ou du graph

Exemples (*publications*) :

- Noeuds : Quelle catégorie de noeuds ?
(laboratoires)
- Arrêtes : Quel lien entre noeuds ?
(relations professionnelles)
- Graph : Quelle type de graph/structure ?
(strategie de publication)



Réseau récurrent

Objectif : apprentissage à partir de séquence ou de série temporel
Domaine : Traitement du signal, compréhension du langage...

Exemple de séquence :

Traduction
Automatique

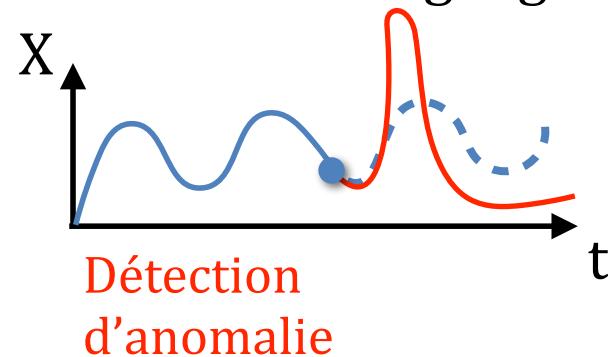
Le film étais très
bien même si il
étais un peu long.

The film was very
good even if it was
a little long.

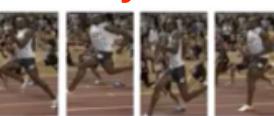
Texte



Classification
d'opinion



Détection
d'anomalie



Course à pied

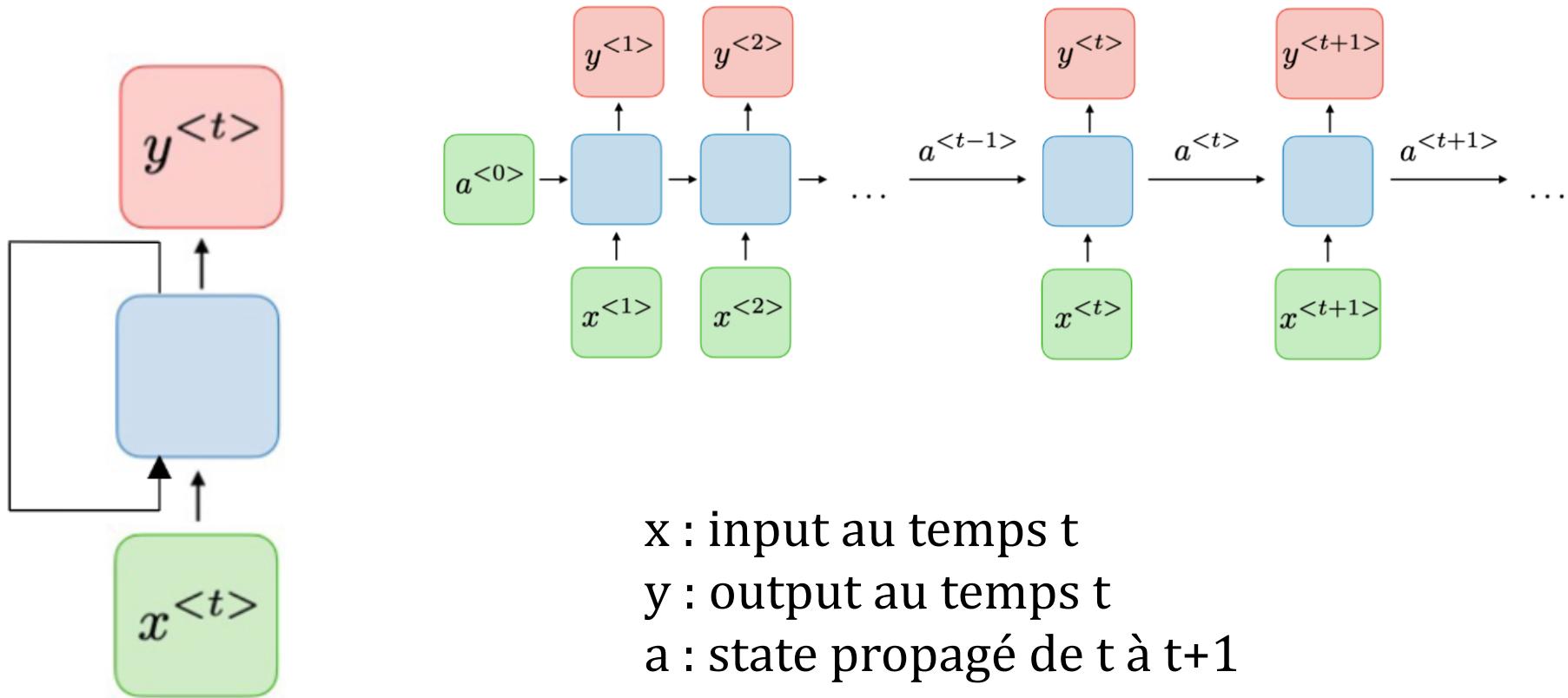


Texte

Reconnaissance de la
parole

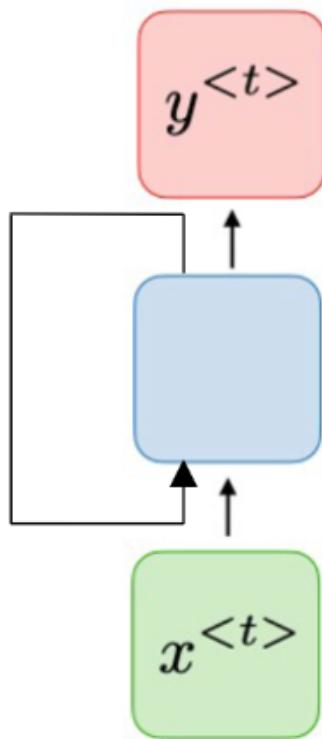
Réseau récurrent

Les réseaux de neurone récurrent (recurrent neural network) sont des réseaux qui utilise la sortie de l'inférence N pour l'inférence N+1

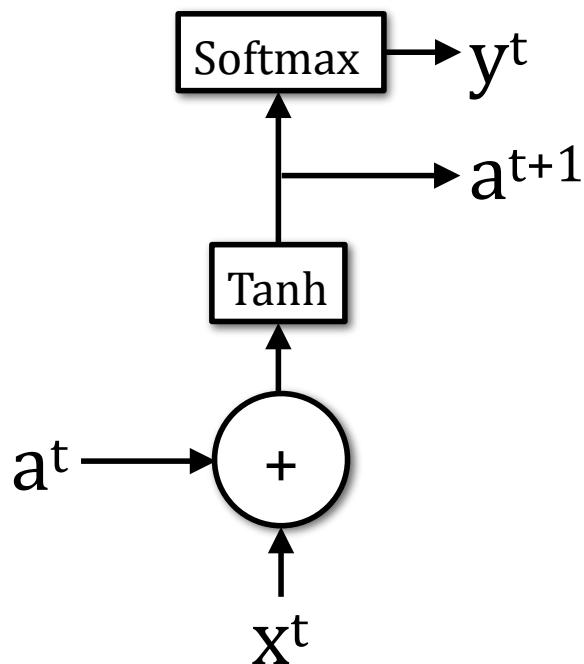


Réseau récurrent : Cell

Différent architecture de RNN existe au niveau des cellules



Cas simple :



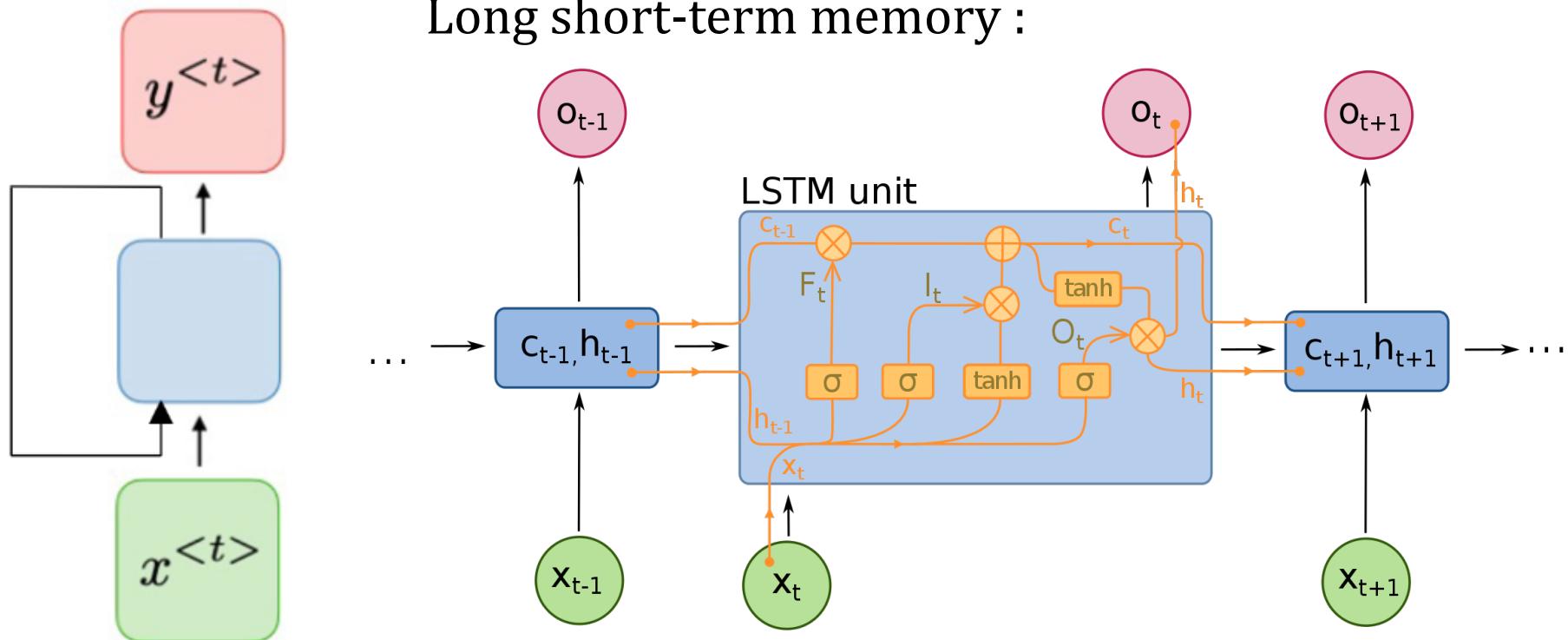
PB : évanouissement/
explosion du gradient

Propagation de gradient
sur le long terme
-> multiplication de
gradient
-> tend vers 0 ou l'infini

Réseau récurrent : Cell

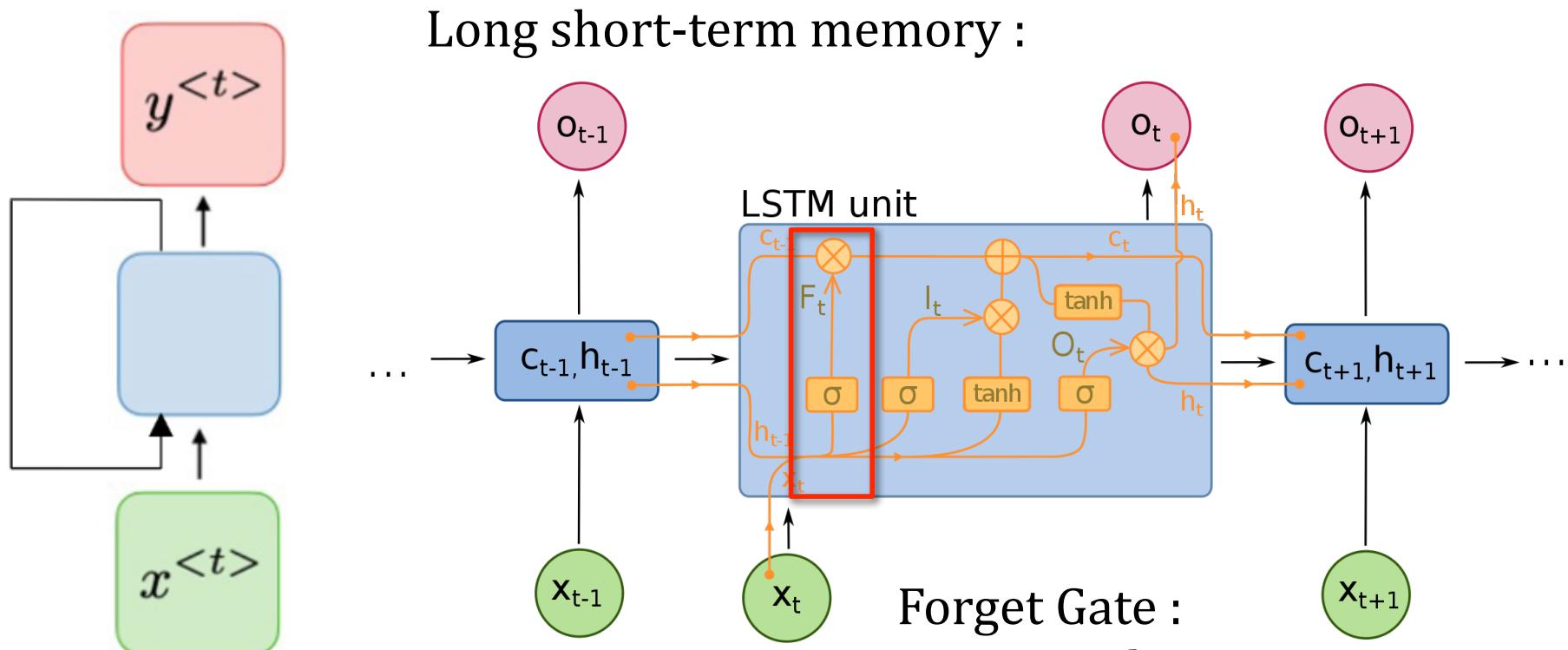
Différent architecture de RNN existe au niveau des cellules

Long short-term memory :



Réseau récurrent : Cell

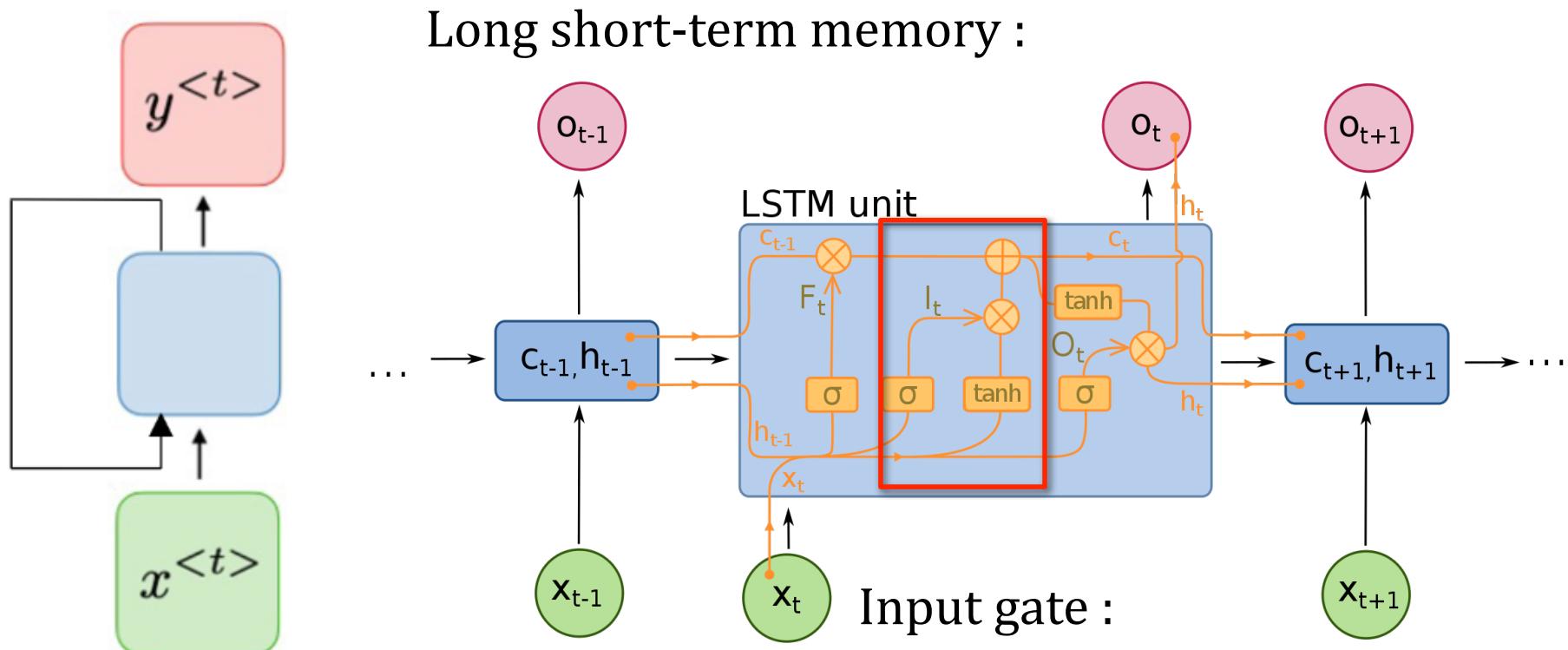
Différent architecture de RNN existe au niveau des cellules



Forget Gate :
supprime les
information non
importante

Réseau récurrent : Cell

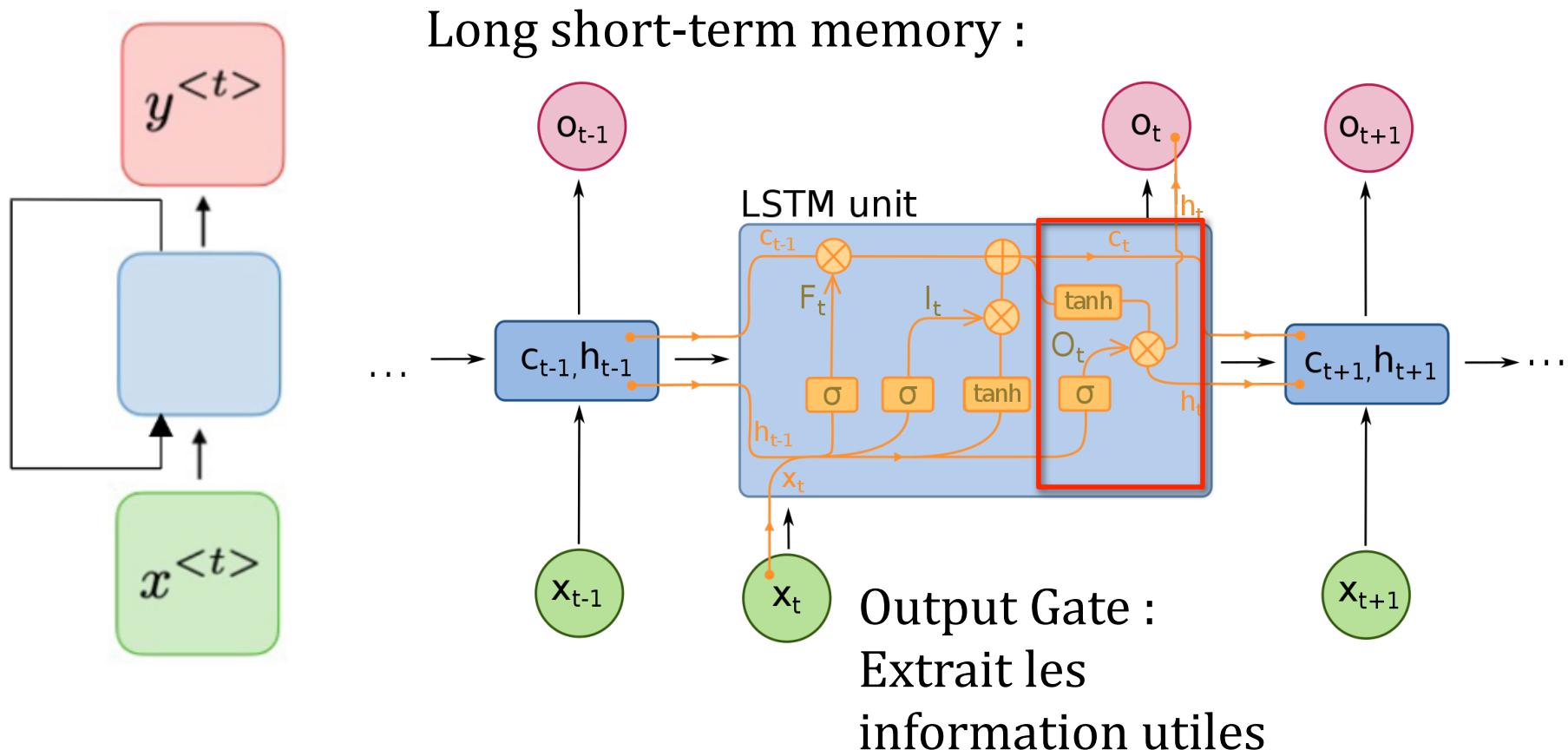
Différent architecture de RNN existe au niveau des cellules



Input gate :
Passe uniquement
les info jugé
importante

Réseau récurrent : Cell

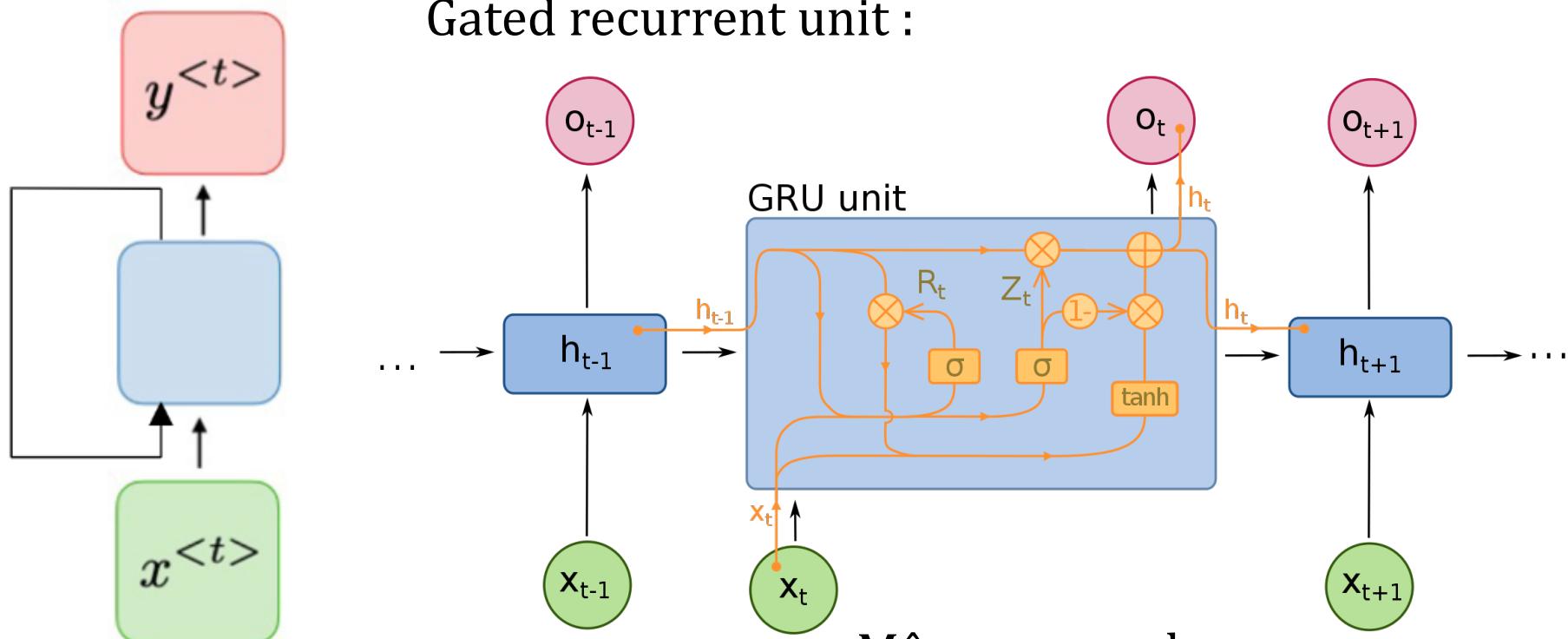
Différent architecture de RNN existe au niveau des cellules



Réseau récurrent : Cell

Différent architecture de RNN existe au niveau des cellules

Gated recurrent unit :



Même approche
sans Output Gate

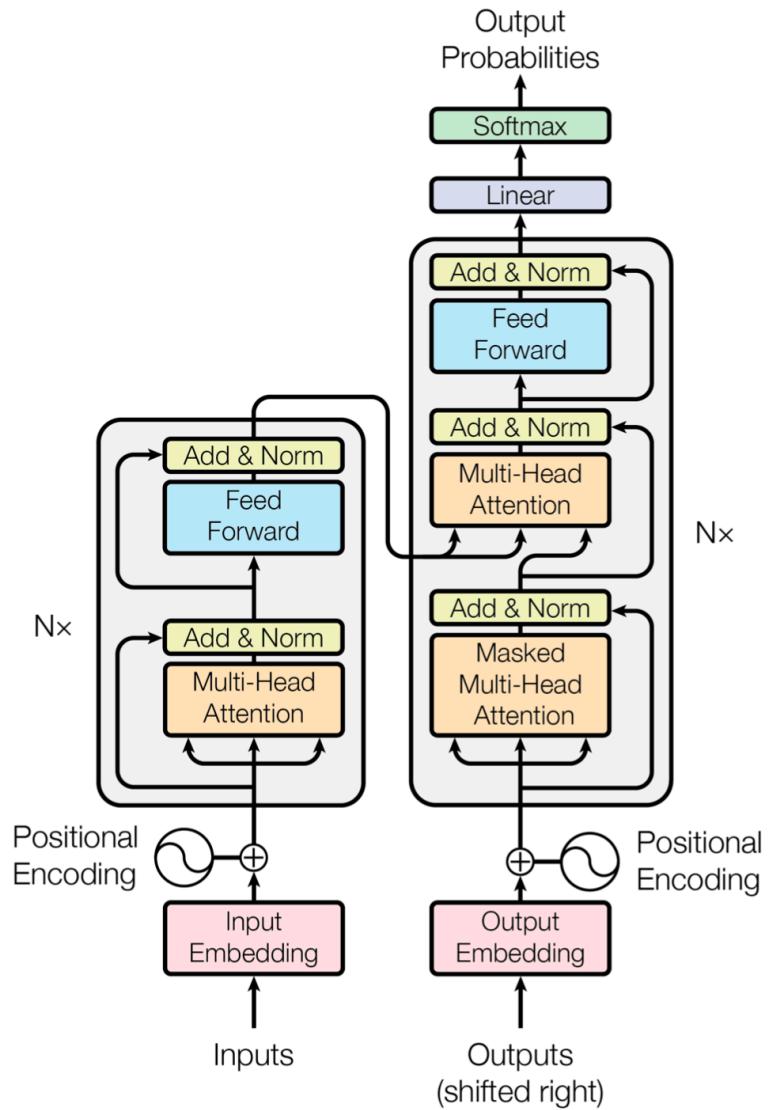
<https://arxiv.org/abs/1412.3555>

Transformer

- Difficulté de memoire dans les RNN -> les réseaux oublient l'information du début
- Solution : Transformer (<https://arxiv.org/abs/1706.03762>)
- Réseaux utilisant un mécanisme d'attention pour garder en mémoire les informations
- ENORME succès : application à la plupart de domaines capables de classification, d'analyse et de génération
- Base de tous les Large Language Model (LLM) dont chatGPT

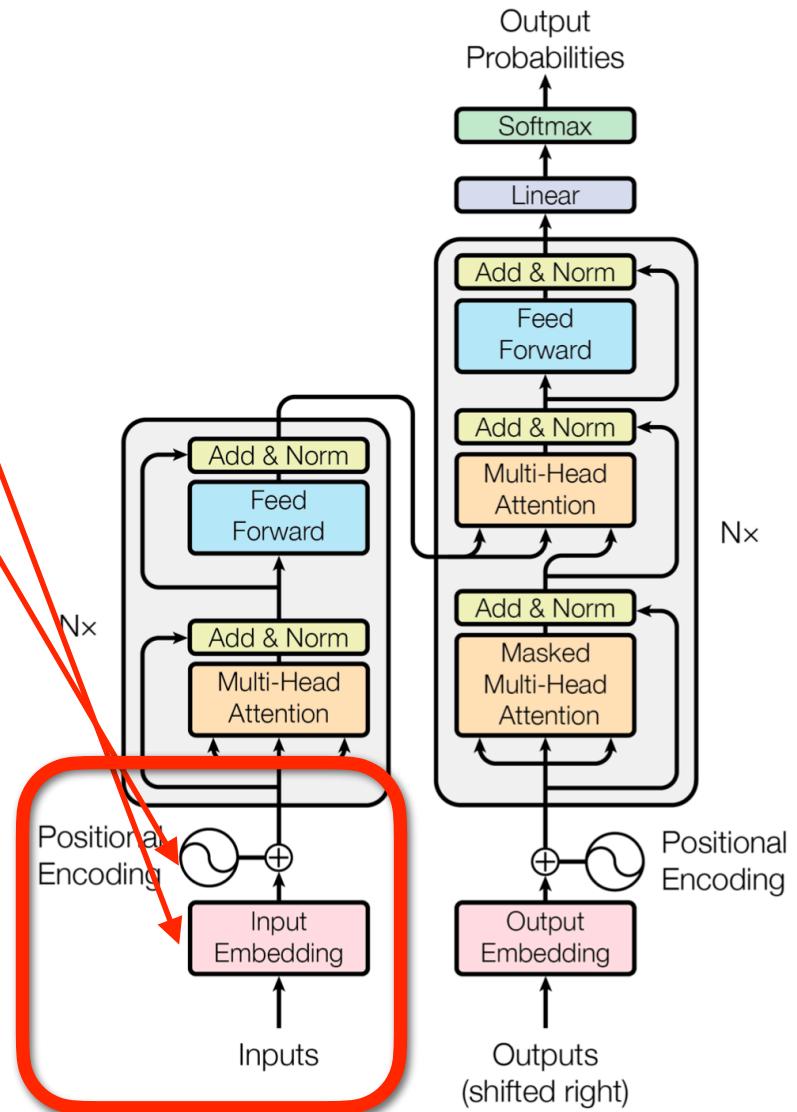
Transformer

- Mécanisme d'attention : apprend les lien entre les différent élément
- Encoding : apprend les liens entre les différent élément de l'input
- Decoding : combine ce qui à déjà été produit avec résultat de l'encoding pour produire un nouvel output



Transformer : Embedding

- Transforme chaque élément de l'input en vecteur taille d :
 x_1, x_2, \dots, x_N
- Dans le cadre du language la position dans la phrase doit aussi être encodé -> modification du vecteur



Transformer : Attention

- Objectif : apprendre les liens qui existe entre les inputs
- Transforme les input en 3 vecteurs de taille d_k par multiplication matriciel :

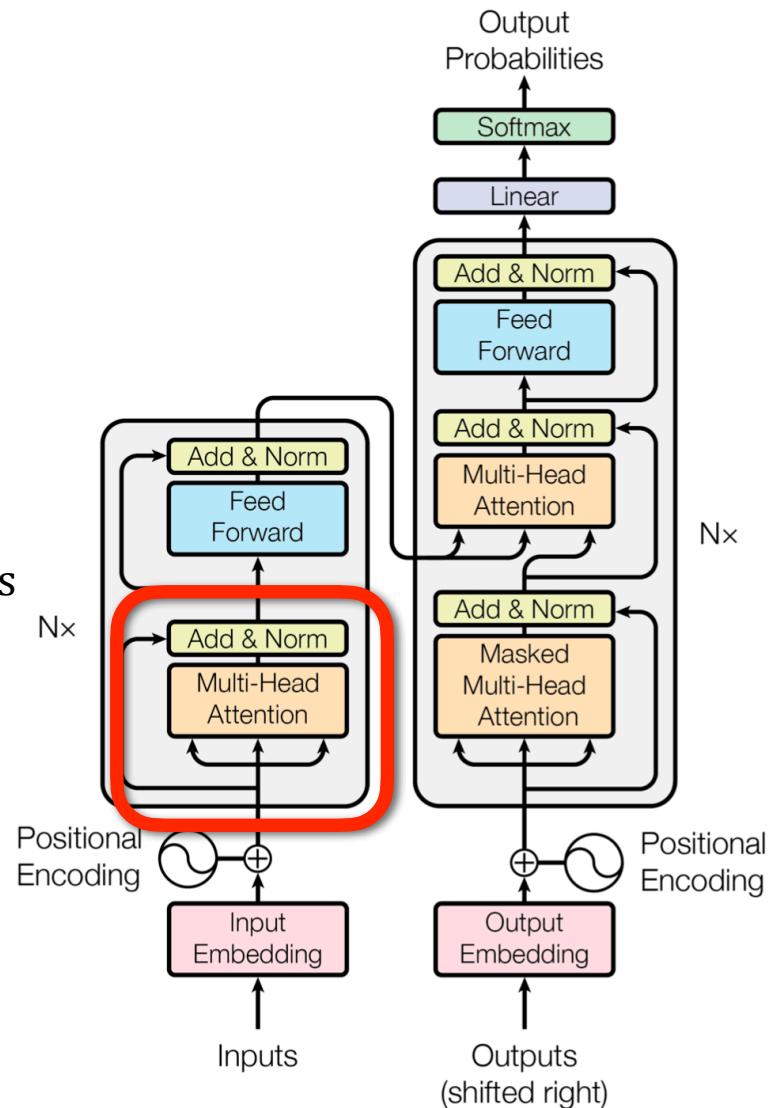
Embedding : x_i

Paramètres appris

$$\text{Queries} : q_i = x_i \times W^Q$$

$$\text{Keys} : k_i = x_i \times W^K$$

$$\text{Values} : v_i = x_i \times W^V$$



Transformer : Attention

Example pour q_2 :

$$score1 = q_2 \times k_1$$

$$score1 = score1 / \sqrt{d}$$

$$score1 = \text{SoftMax}(score1) = 0.9$$

$$score1 = score1 \cdot v_1$$

$$score2 = q_2 \times k_2$$

$$score2 = score2 / \sqrt{d}$$

$$score2 = \text{SoftMax}(score2) = 0.1$$

$$score2 = score2 \cdot v_2$$

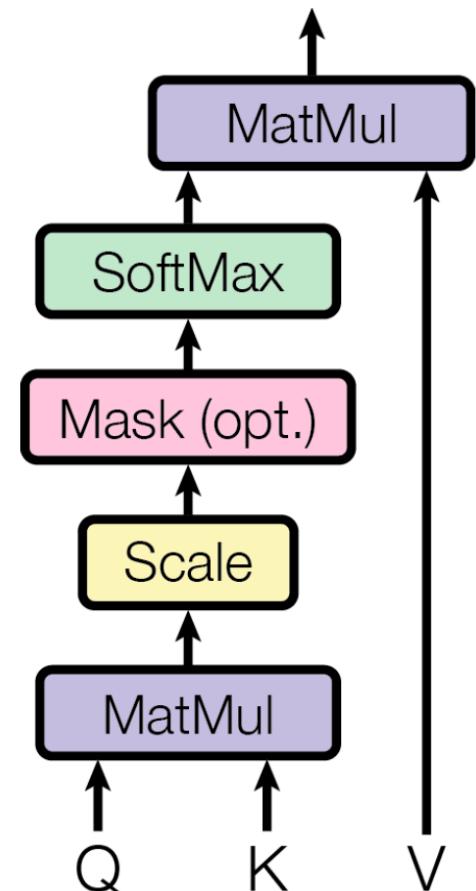
...

$$z_2 = \sum_{I=1}^N scoreI$$

Fort lien

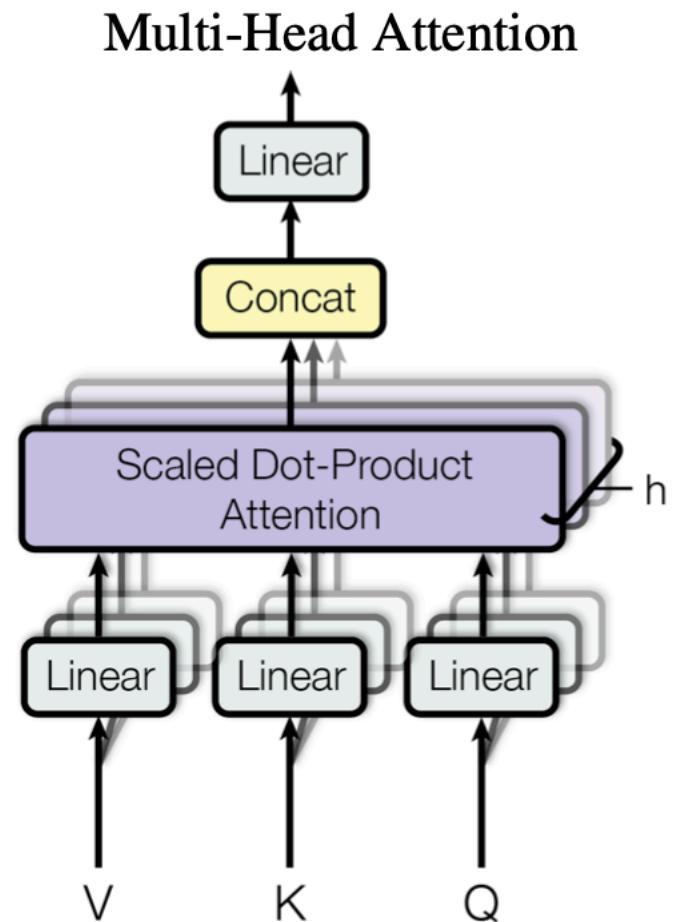
Faible lien

sortie de taille N



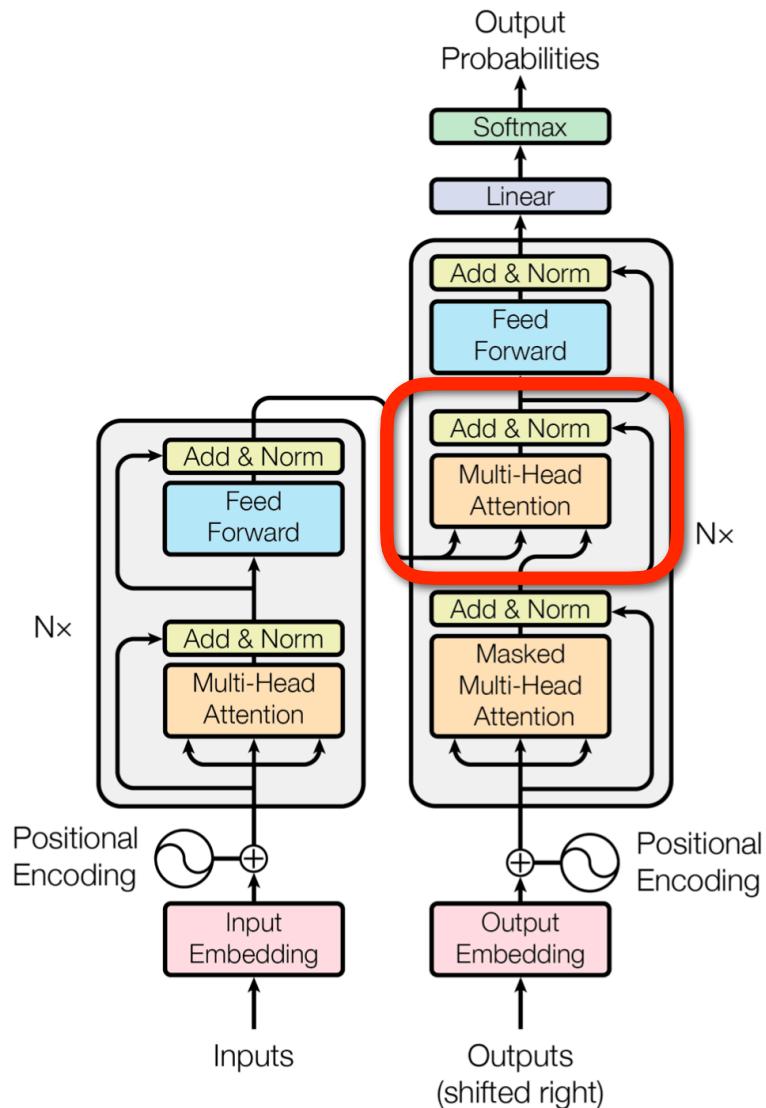
Transformer : Attention

- Opération effectuer h fois en parallèle tel que : $d = h \times d_k$
- Plus efficace et rapide
- Résultat combiné a-posteriori



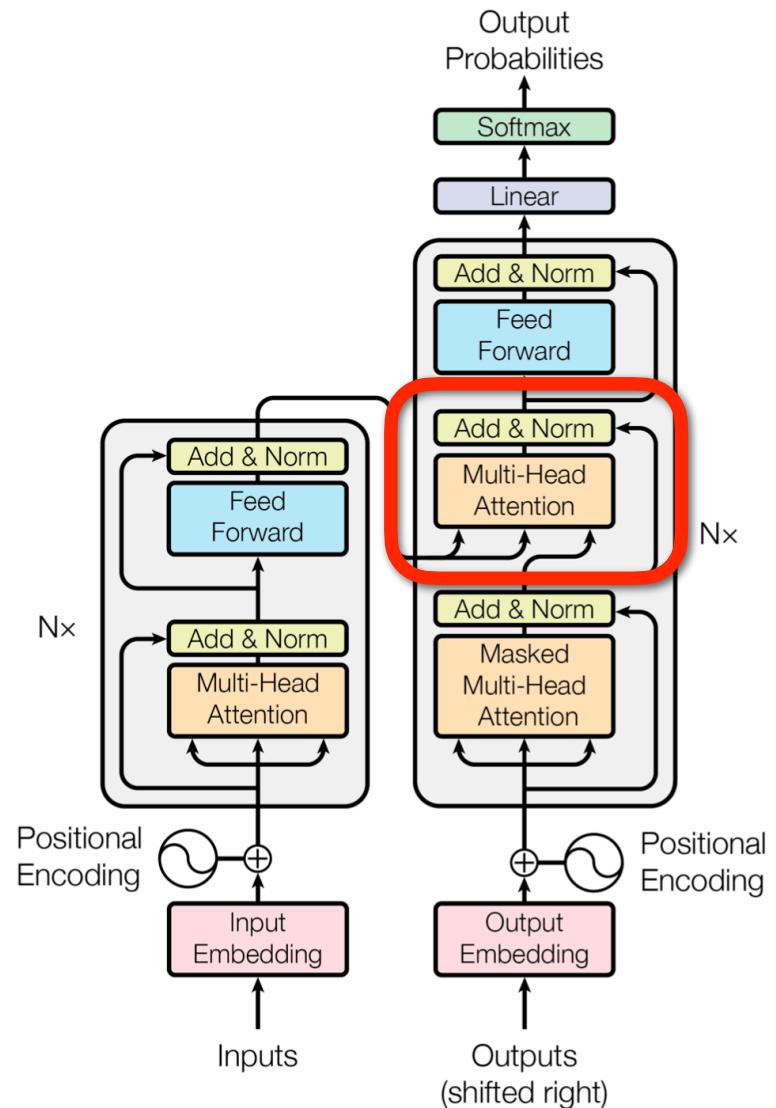
Transformer : Décodeur

- Après l'attention passage dans un NN
- Dans le décodeur les queries q' des output précédent sont comparé aux keys et values de l'encodeur
- L'opération se répète jusqu'à ce que l'output corresponde au caractère de fin d'opération



Transformer : Exemple

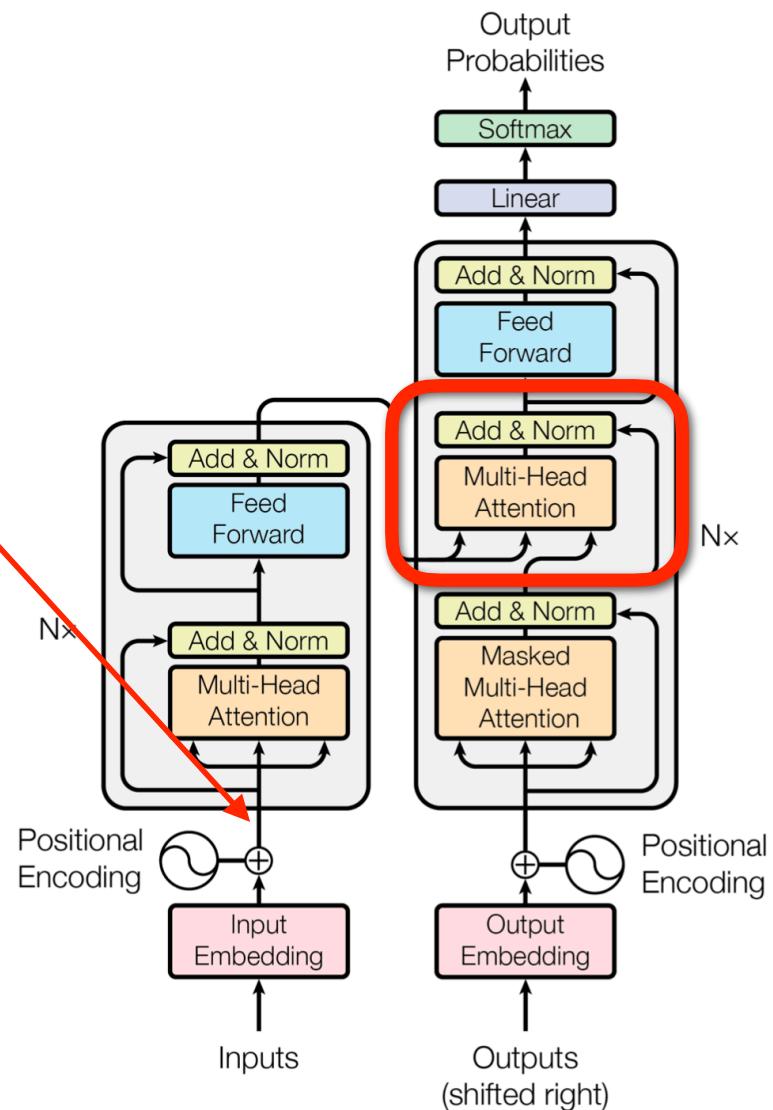
Traduction : « Je mange un kiwi »



Transformer : Exemple

Traduction : « Je mange un kiwi »

Embedding : $[x_1, x_2, x_3, x_4]$

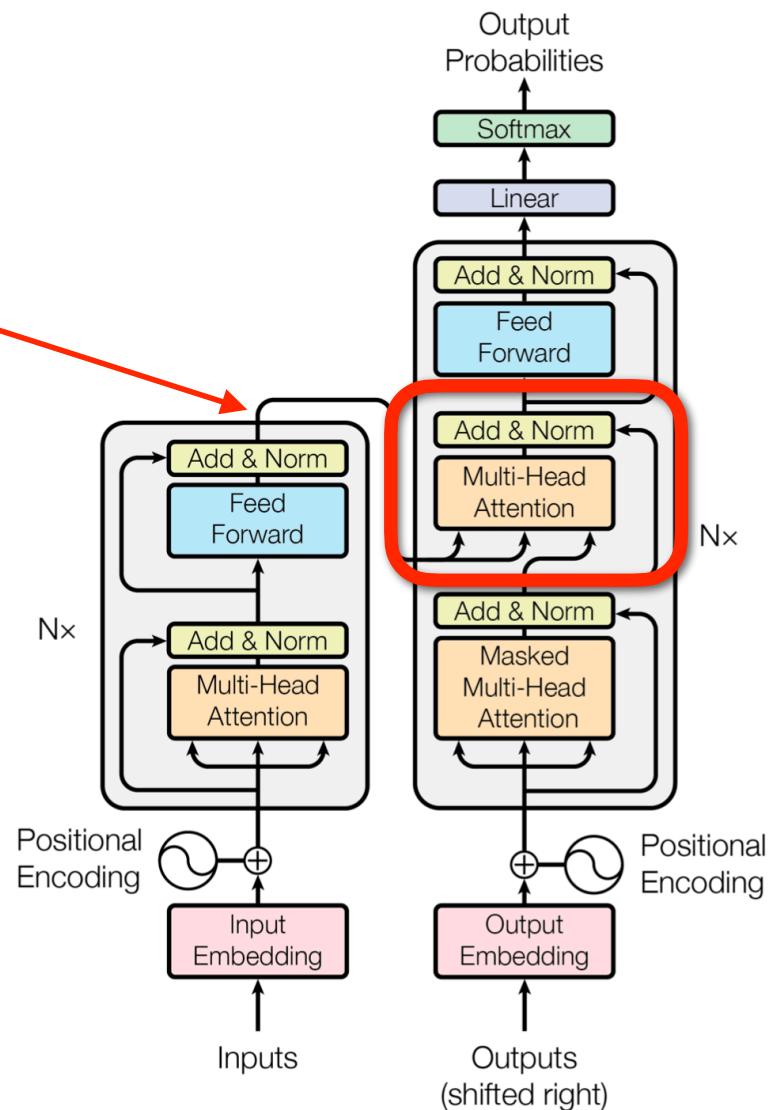


Transformer : Exemple

Traduction : « Je mange un kiwi »

Embedding : $[x_1, x_2, x_3, x_4]$

N boucle d'encodage : $[z_1, z_2, z_3, z_4]$



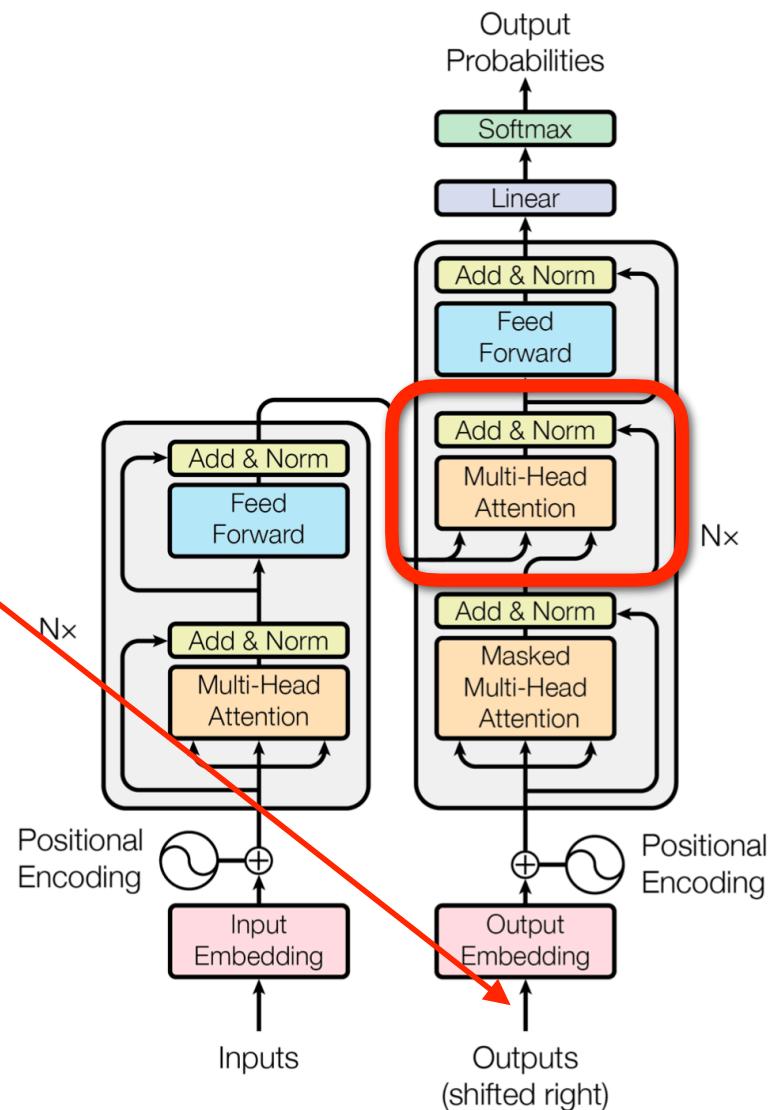
Transformer : Exemple

Traduction : « Je mange un kiwi »

Embedding : $[x_1, x_2, x_3, x_4]$

N boucle d'encodage : $[z_1, z_2, z_3, z_4]$

Décodage : Output = $\emptyset \rightarrow o_0$



Transformer : Exemple

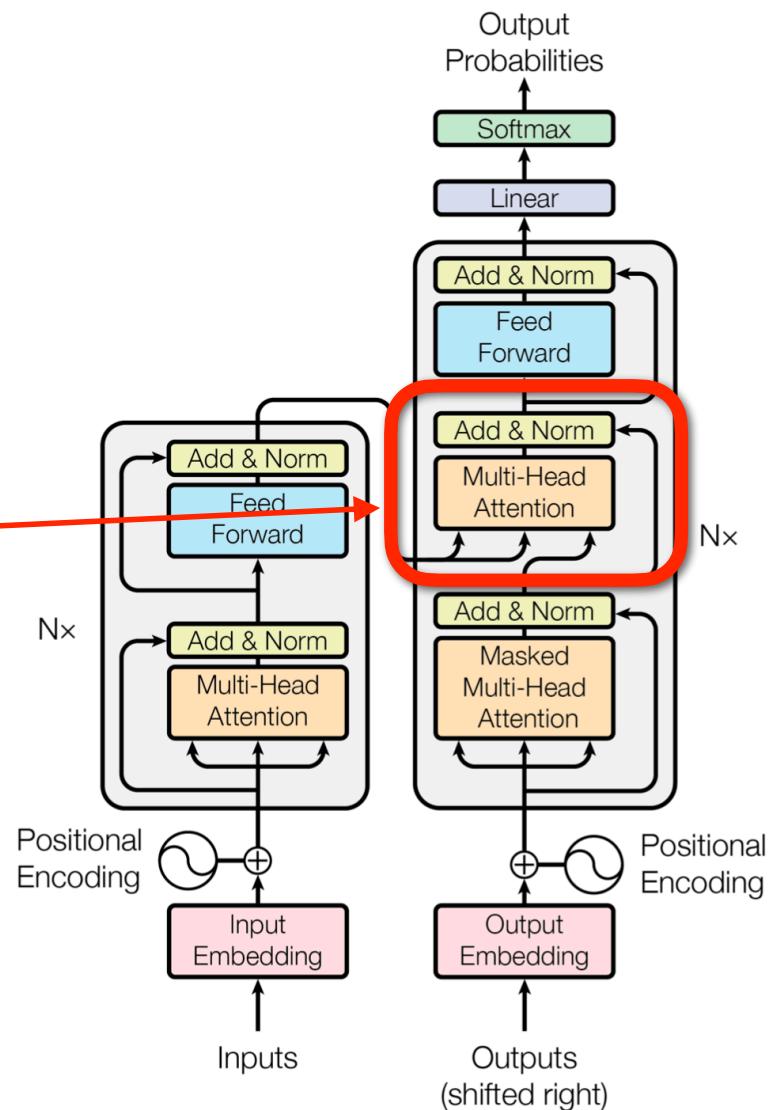
Traduction : « Je mange un kiwi »

Embedding : $[x_1, x_2, x_3, x_4]$

N boucle d'encodage : $[z_1, z_2, z_3, z_4]$

Décodage : Output = $\emptyset \rightarrow o_0$

Décodage : combine $[z_1, z_2, z_3, z_4]$ et les o_i



Transformer : Exemple

Traduction : « Je mange un kiwi »

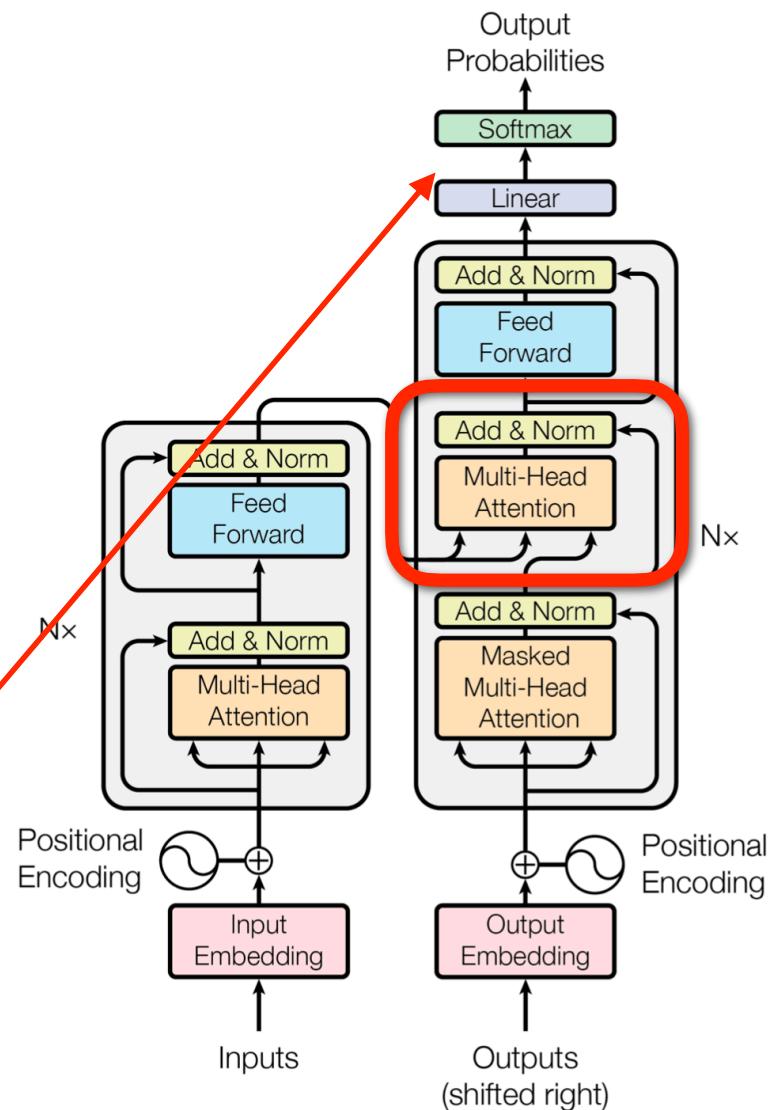
Embedding : $[x_1, x_2, x_3, x_4]$

N boucle d'encodage : $[z_1, z_2, z_3, z_4]$

Décodage : Output = $\emptyset \rightarrow o_0$

Décodage : combine $[z_1, z_2, z_3, z_4]$
et les o_i

Output : o'_1 probabilité des différents
mots \rightarrow Sélectionne le max : $o_1 = I$



Transformer : Exemple

Traduction : « Je mange un kiwi »

Embedding : $[x_1, x_2, x_3, x_4]$

N boucle d'encodage : $[z_1, z_2, z_3, z_4]$

Décodage : Output = $\emptyset \rightarrow o_0$

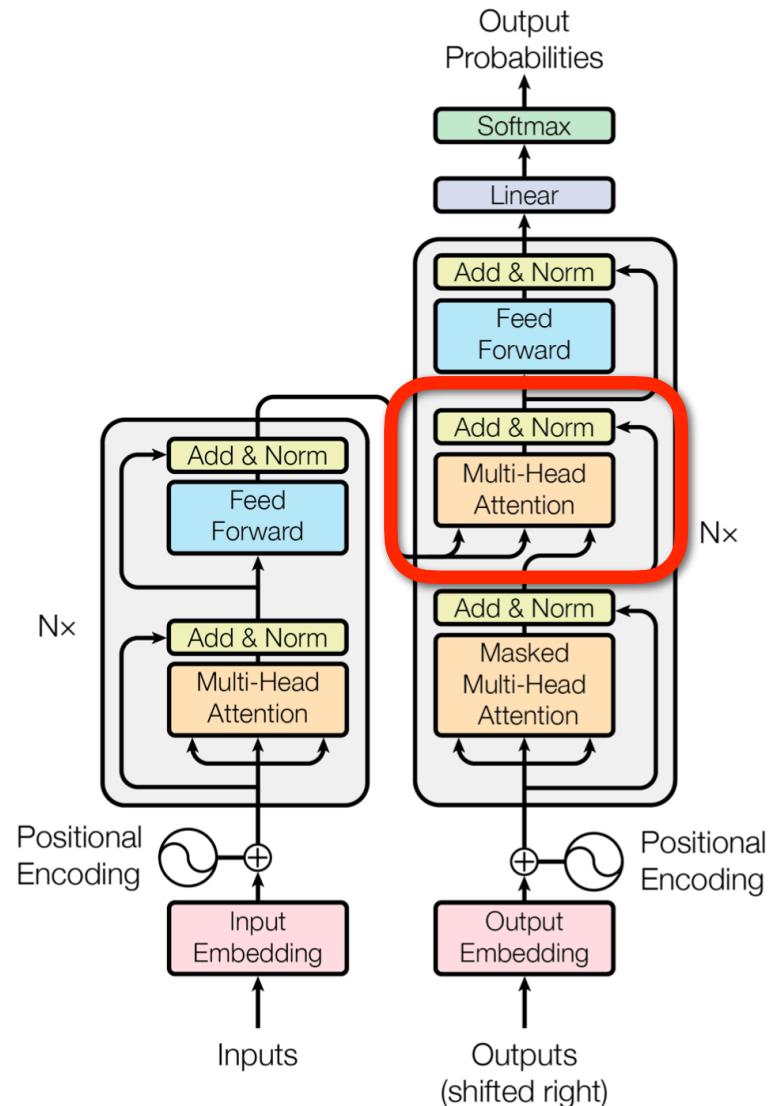
Décodage : combine $[z_1, z_2, z_3, z_4]$
et les o_i

Output : o'_1 probabilité des différents
mots \rightarrow Sélectionne le max : $o_1 = I$

Continue le décodage :

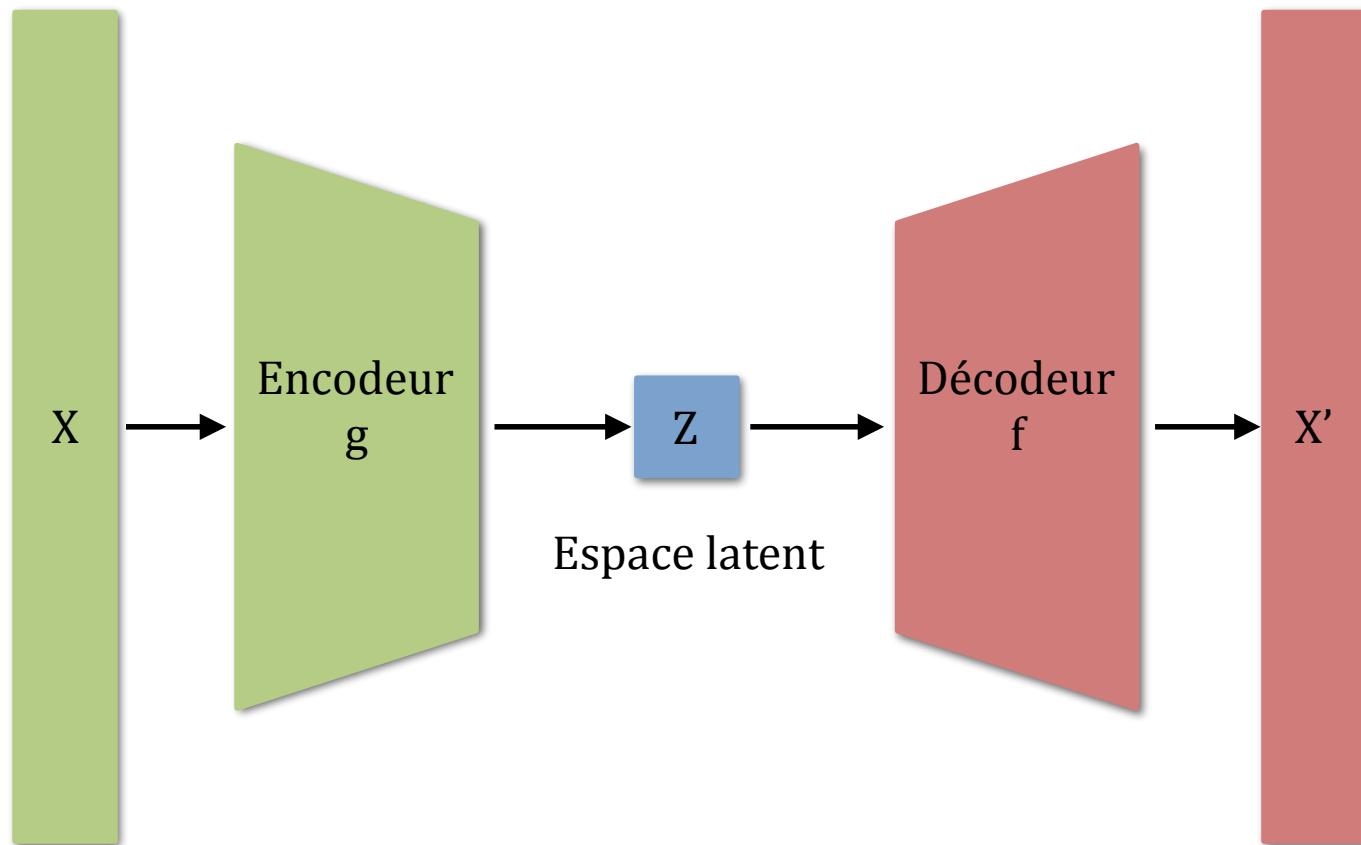
$o_1 \rightarrow o_2 = am$ $[o_1, o_2, o_3] \rightarrow o_4 = a$

$[o_1, o_2] \rightarrow o_3 = eating$ $[o_1, o_2, o_3, o_4] \rightarrow o_5 = kiwi$
 $[o_1, o_2, o_3, o_4, o_5] \rightarrow o_6 = <eos>$



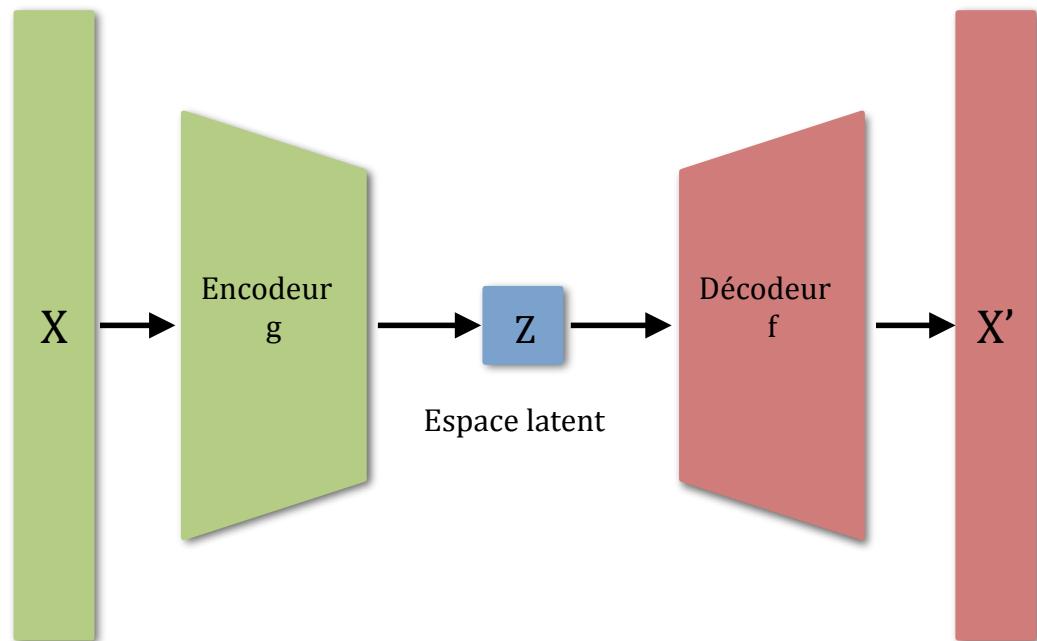
Auto Encoders

Application : Débruitage, réduction de dimension, détection d'anomalie



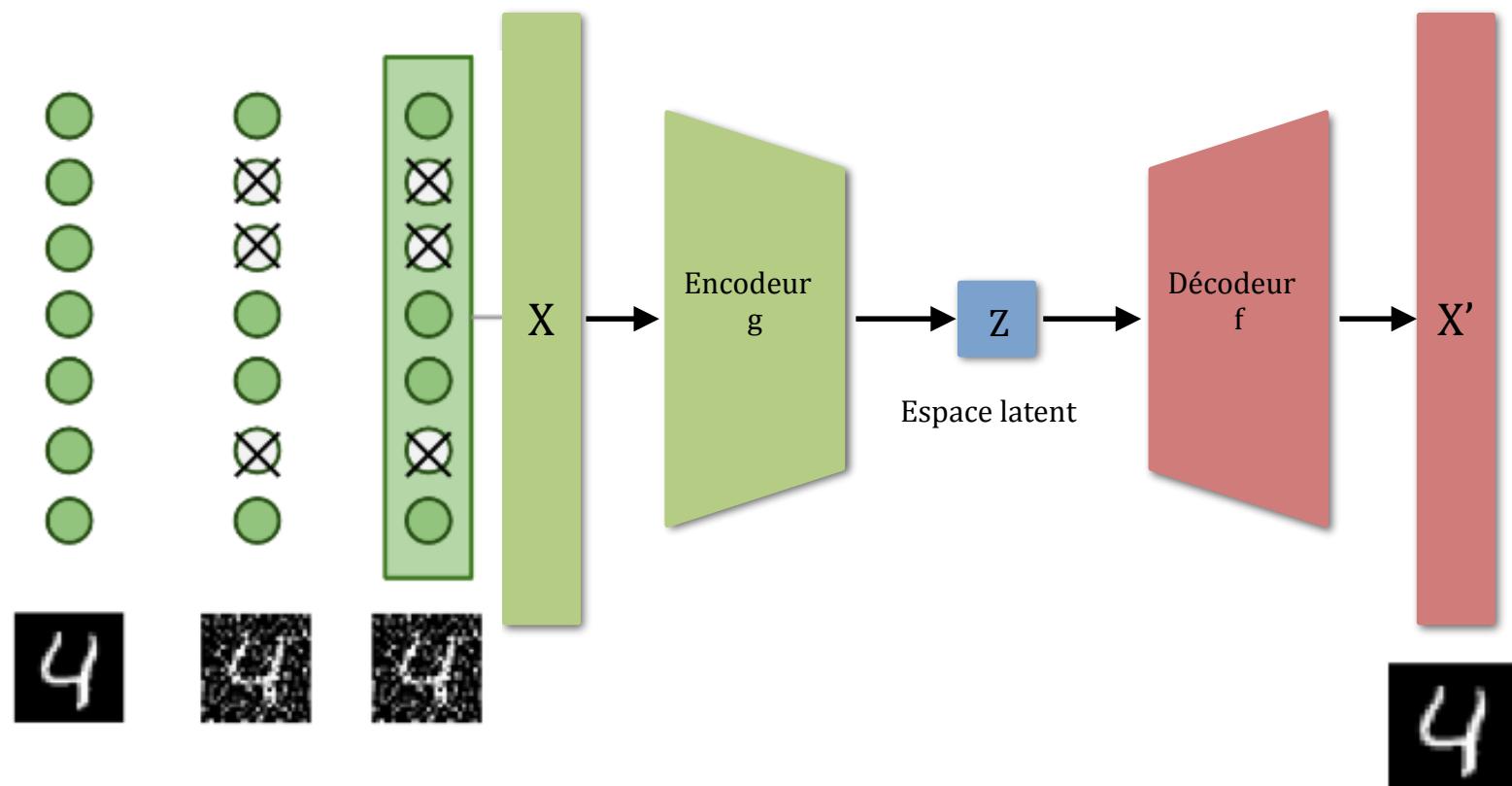
Auto Encoders

- Encodeur : réduit la taille de l'entrée en un petit espace latent
- Décodeur : reconstruit l'entrée à partir de l'espace latent
- Fonction de coût minimise la différence entre X' et X



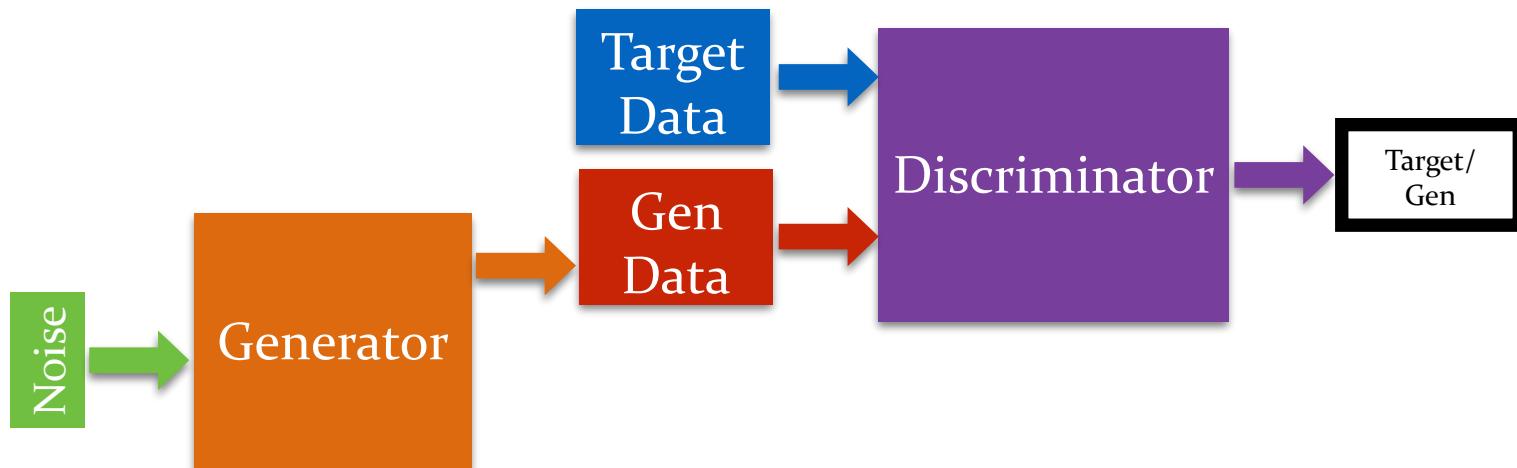
Auto Encoders : debruitage

Même espace latent avec
des entré bruité



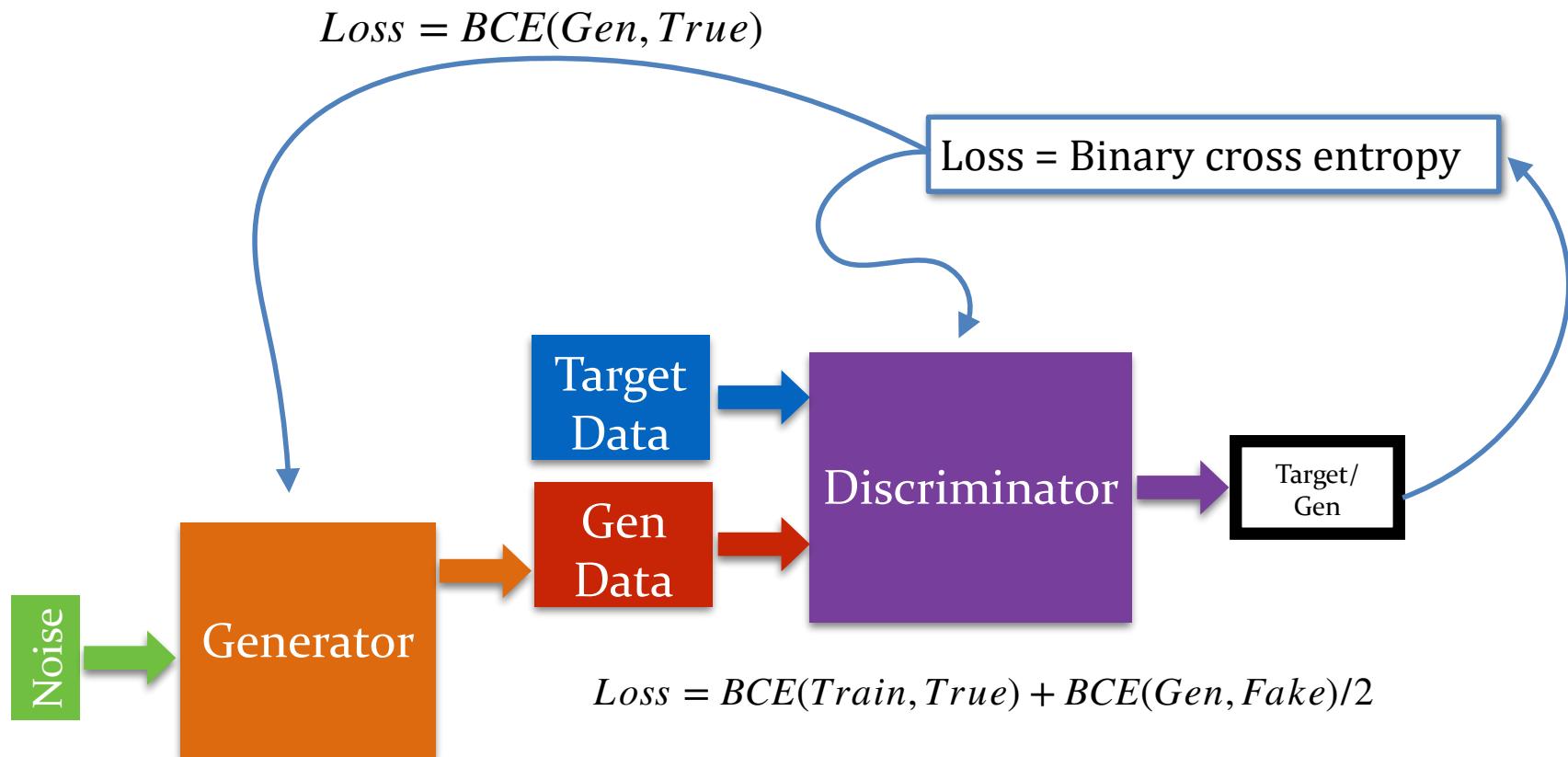
Generative Adversarial Network (GAN)

- Technique générative : crée des données/images à partir de bruit
- Utilise deux réseaux de neurones :
 - Générateur : génère des donnée à partir de bruit
 - Discriminateur : sépare le bruit des donnée cible



Generative Adversarial Network (GAN)

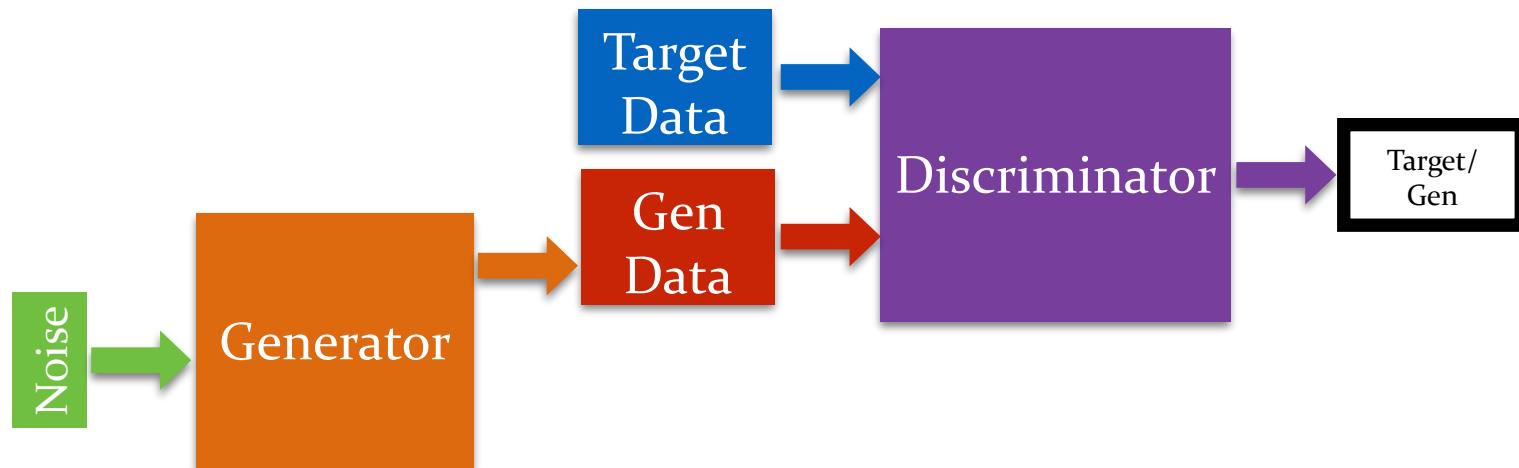
Même fonction de cout pour le discriminateur et le générateur



Generative Adversarial Network (GAN)

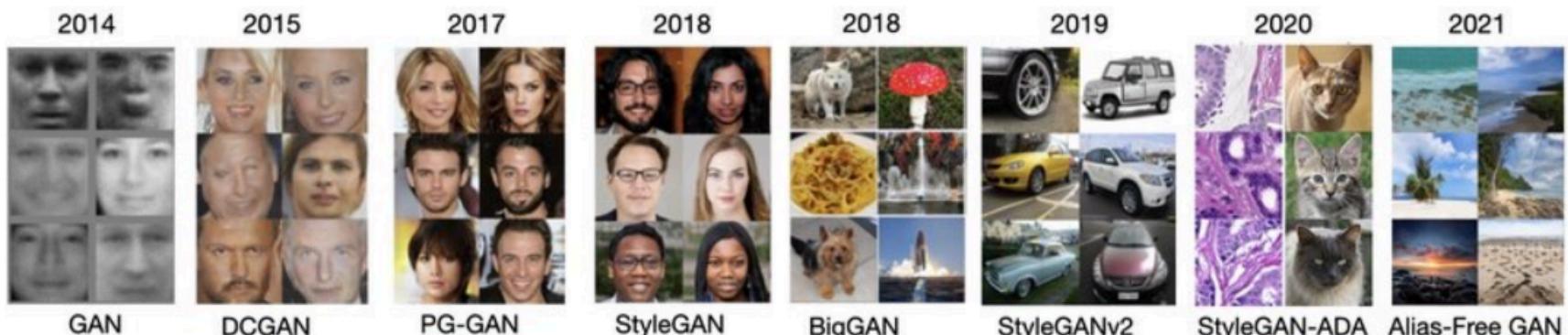
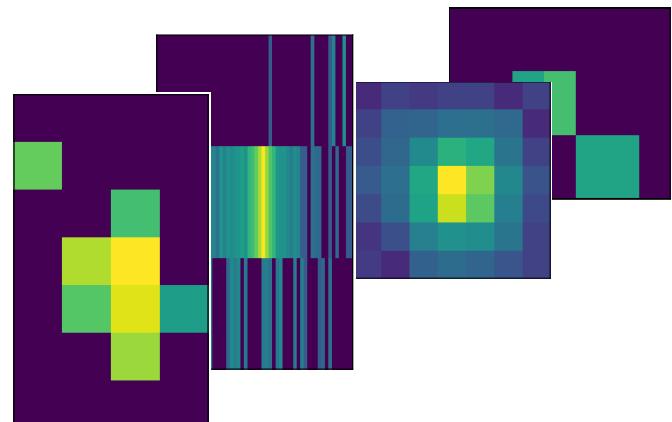
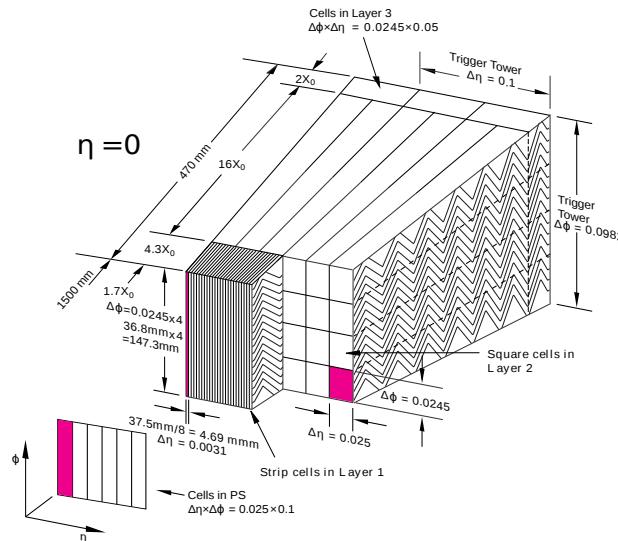
Le type de NN utilisé pour le générateur peuvent varié :

- Perceptron multi-couche
- Convolutional NN
- ...



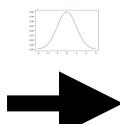
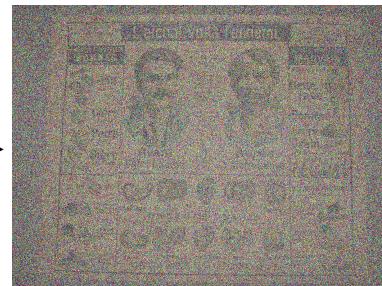
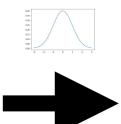
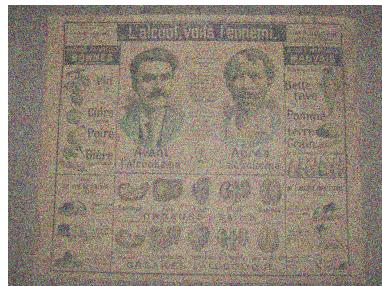
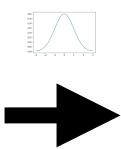
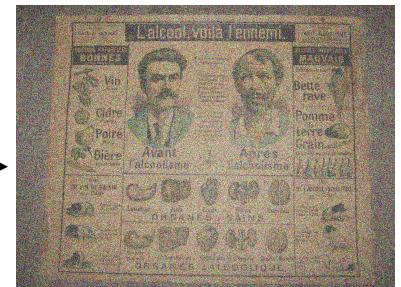
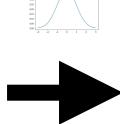
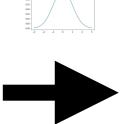
Generative Adversarial Network (GAN)

Utilisation : Simulation, génération d'image, ...

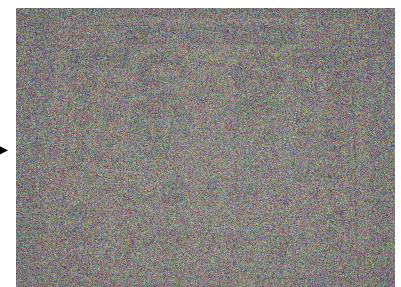
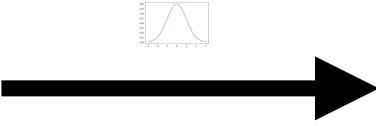


Diffusion models

Approche similaire au AE mais appliquée à la génération



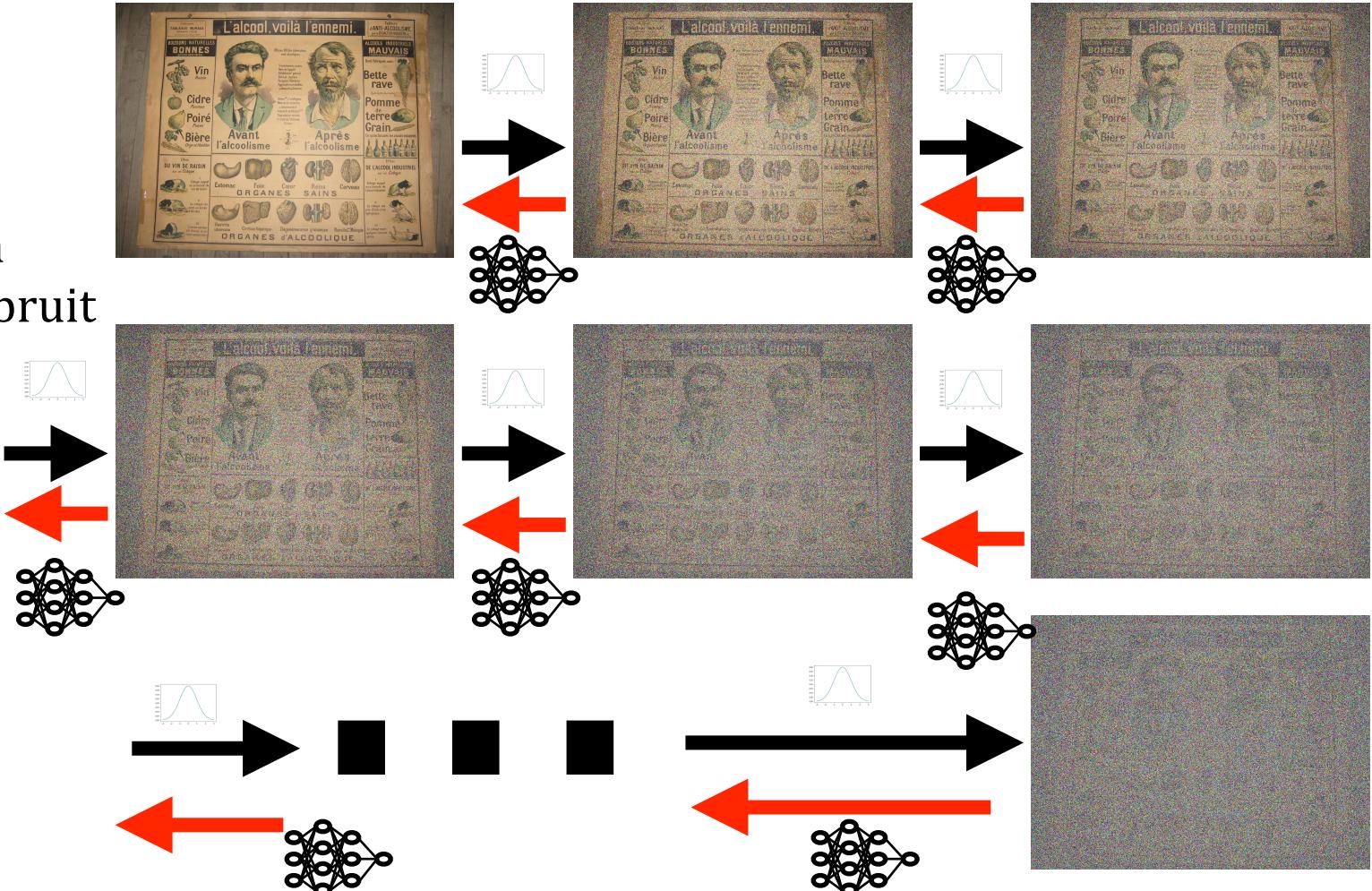
Ajout successif
de petit bruit
gaussien



Diffusion models

Approche similaire au AE mais appliquée à la génération

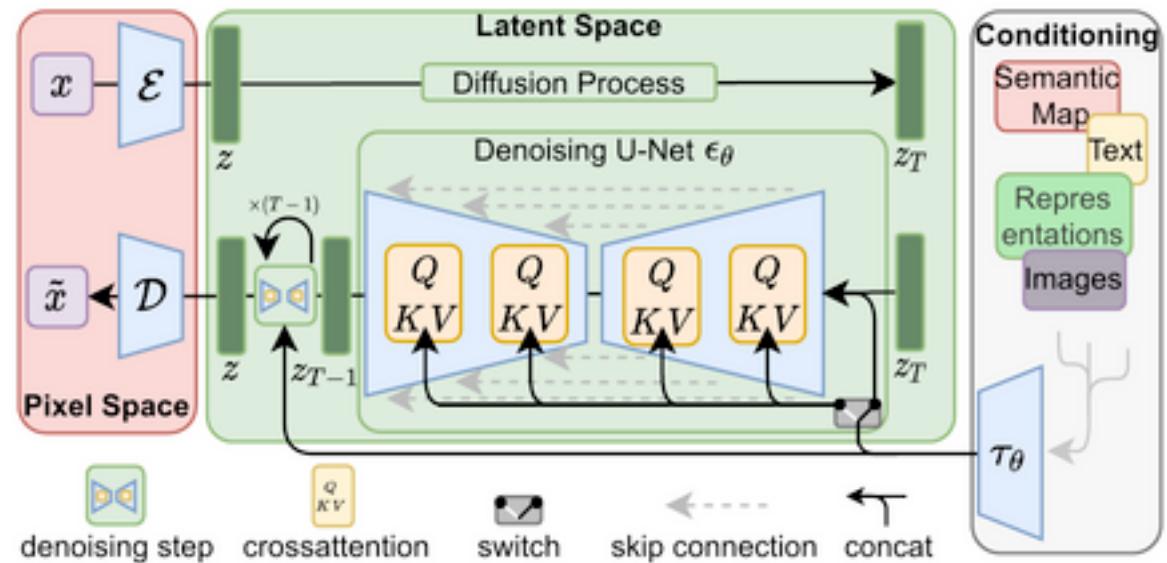
NN qui apprend à retirer le bruit gaussien



Diffusion models

Peut partir du bruit pour reconstruire des images

Possibilité d'ajouté des condition à la reconstruction



BACKUP