

Réseaux de neurones

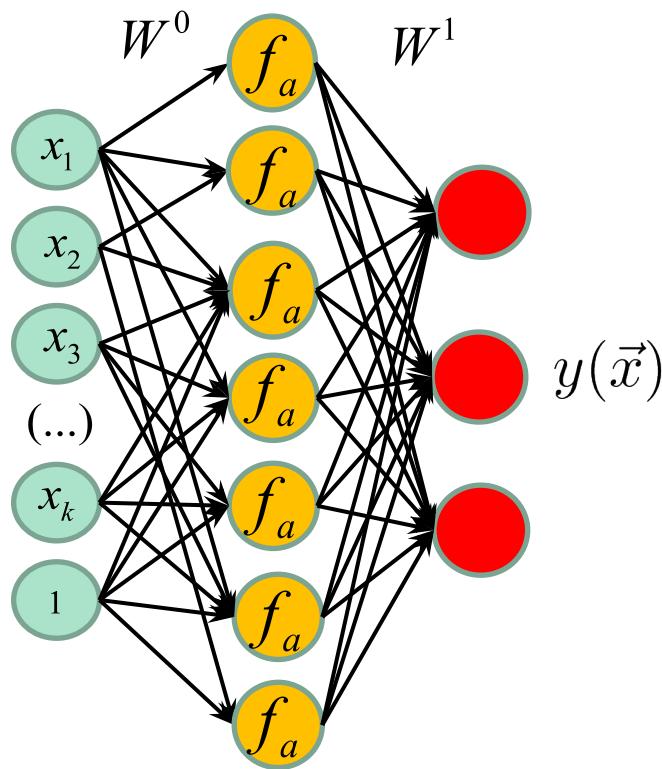
IFT 780

Réseaux récurrents

Par

Pierre-Marc Jodoin, Antoine Théberge

Réseau de neurones de base (régression)



$$y(\vec{x}) = W^1 f_a(W^0 \vec{x})$$

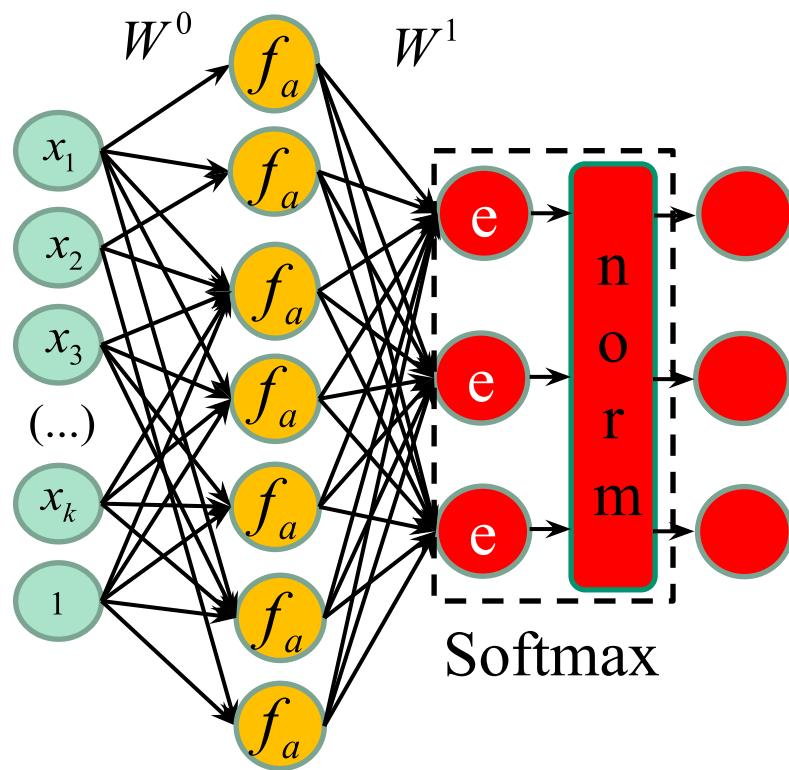


$$\vec{h} = f_a(W^0 \vec{x})$$

$$y(\vec{x}) = W^1 \vec{h}$$

f_a : fonction d'activation

Réseau de neurones de base (classification)



$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0 \vec{x}))$$



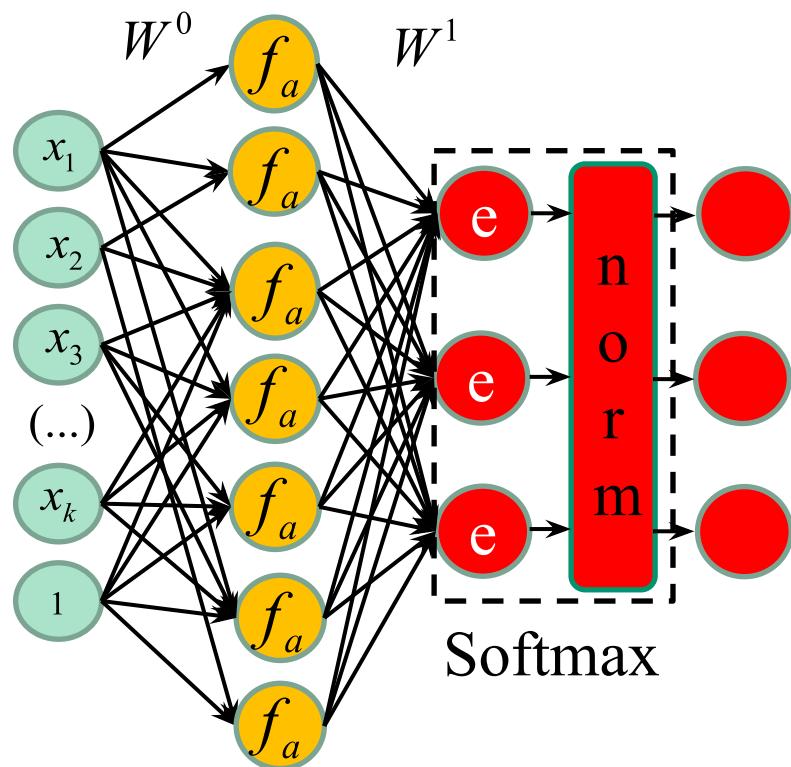
$$y(\vec{x})$$

$$\vec{h} = f_a(W^0 \vec{x})$$

$$\hat{y} = W^1 \vec{h}$$

$$y(\vec{x}) = \text{softmax}(\hat{y})$$

Réseau de neurones de base (classification)



$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0 \vec{x}))$$



$$y(\vec{x})$$

$$\vec{h} = f_a(W^0 \vec{x})$$

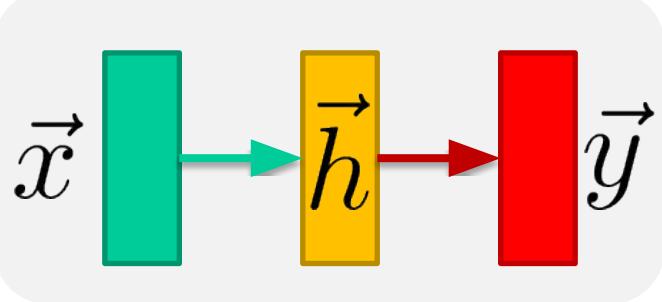
$$\hat{y} = W^1 \vec{h}$$

$$y(\vec{x}) = \text{softmax}(\hat{y})$$

Ne permettent que des tâches “1 pour 1”

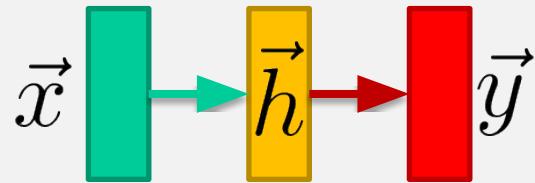
- Classification (1 image = 1 label)
- Régression (1 donnée = 1 chiffre)
- Localisation (1 boîte = 1 classification + 1 régression)

Illustration simplifiée



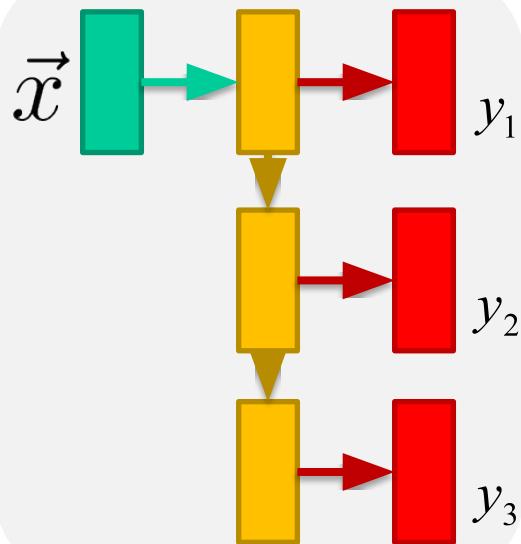
RN de base

Différentes configurations pour différentes applications



RN de base

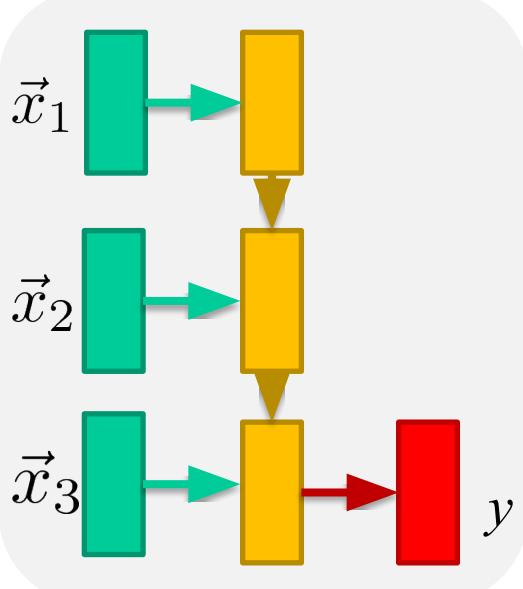
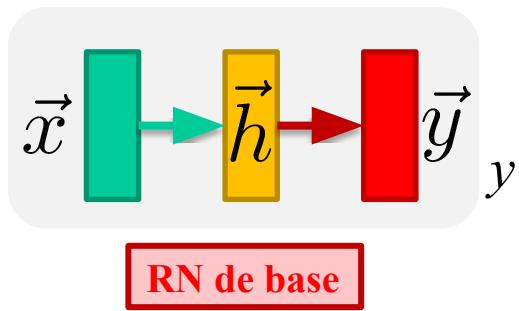
1 entrée et N sorties



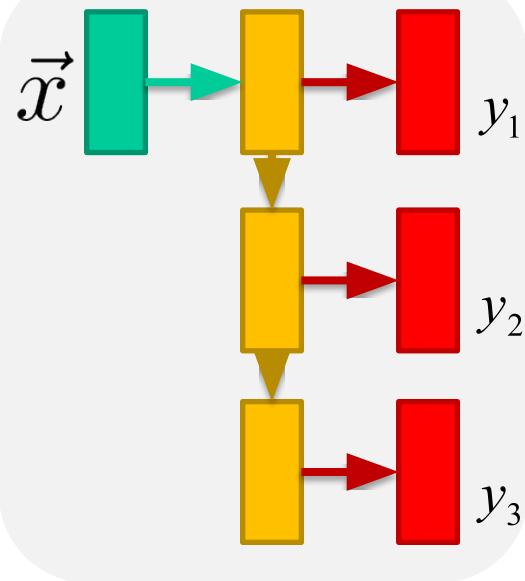
Ex.: description d'une image
1 image => N mots

Différentes configurations pour différentes applications

1 entrée et 1 sorties



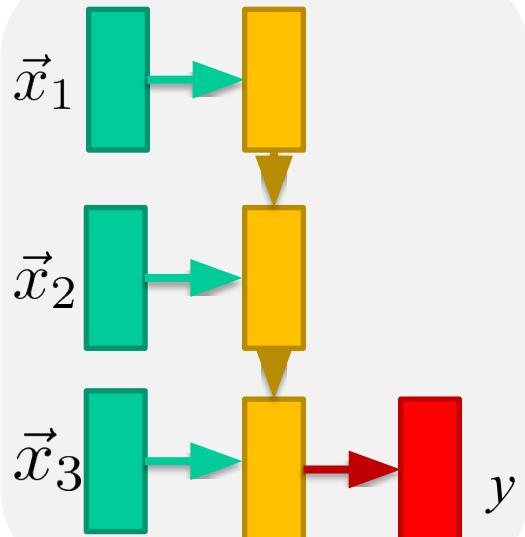
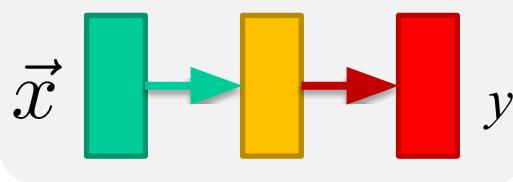
1 entrée et N sorties



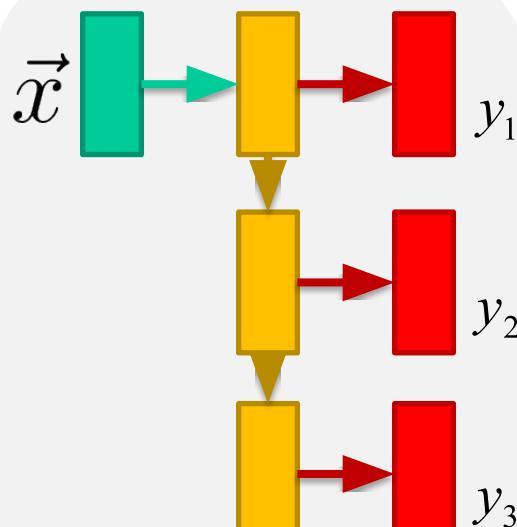
Différentes configurations pour différentes applications

N entrées et 1 sortie

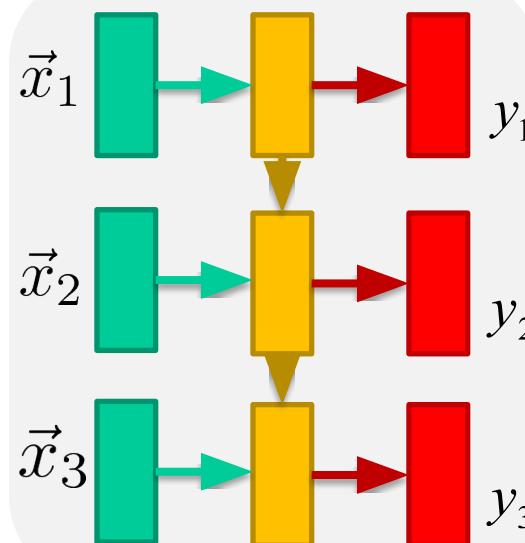
1 entrée et 1 sortie



1 entrée et N sorties



N entrées et N sorties



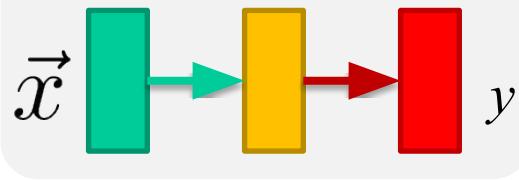
Ex.: Classification d'images vidéo
N images => N prédictions

Différentes configurations pour différentes applications

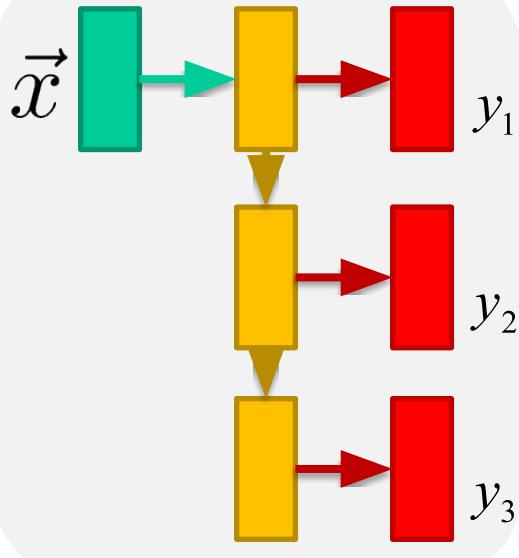
M entrées et N sorties

1 entrée et N sorties

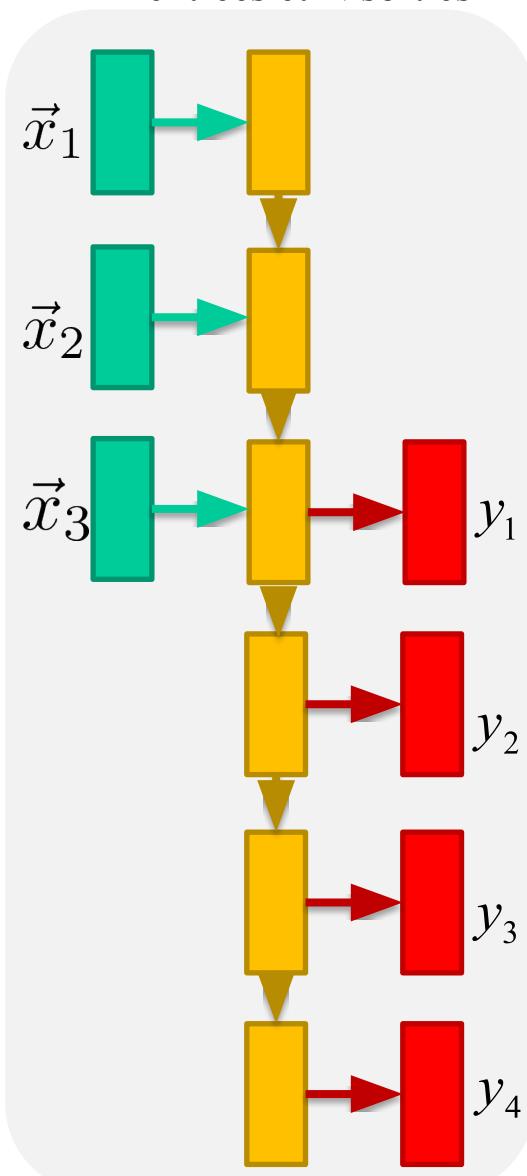
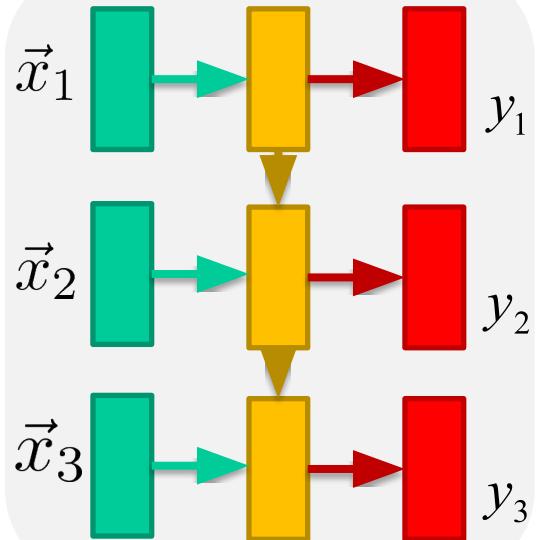
1 entrée et 1 sortie



1 entrée et N sorties

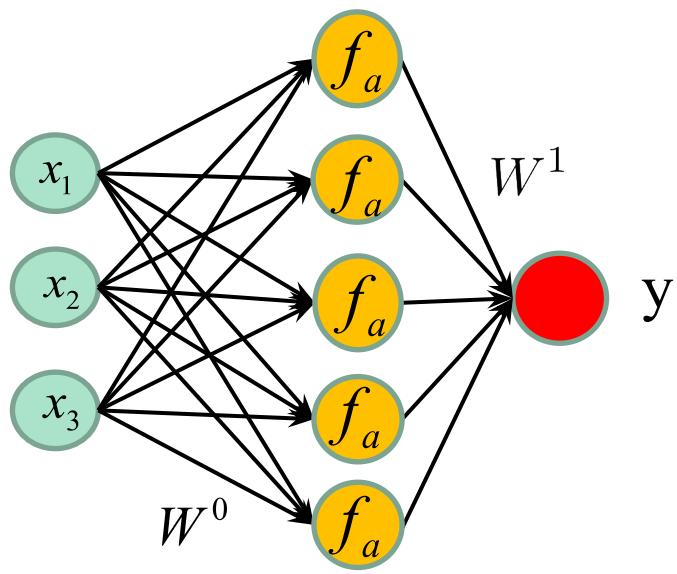


N entrées et N sorties

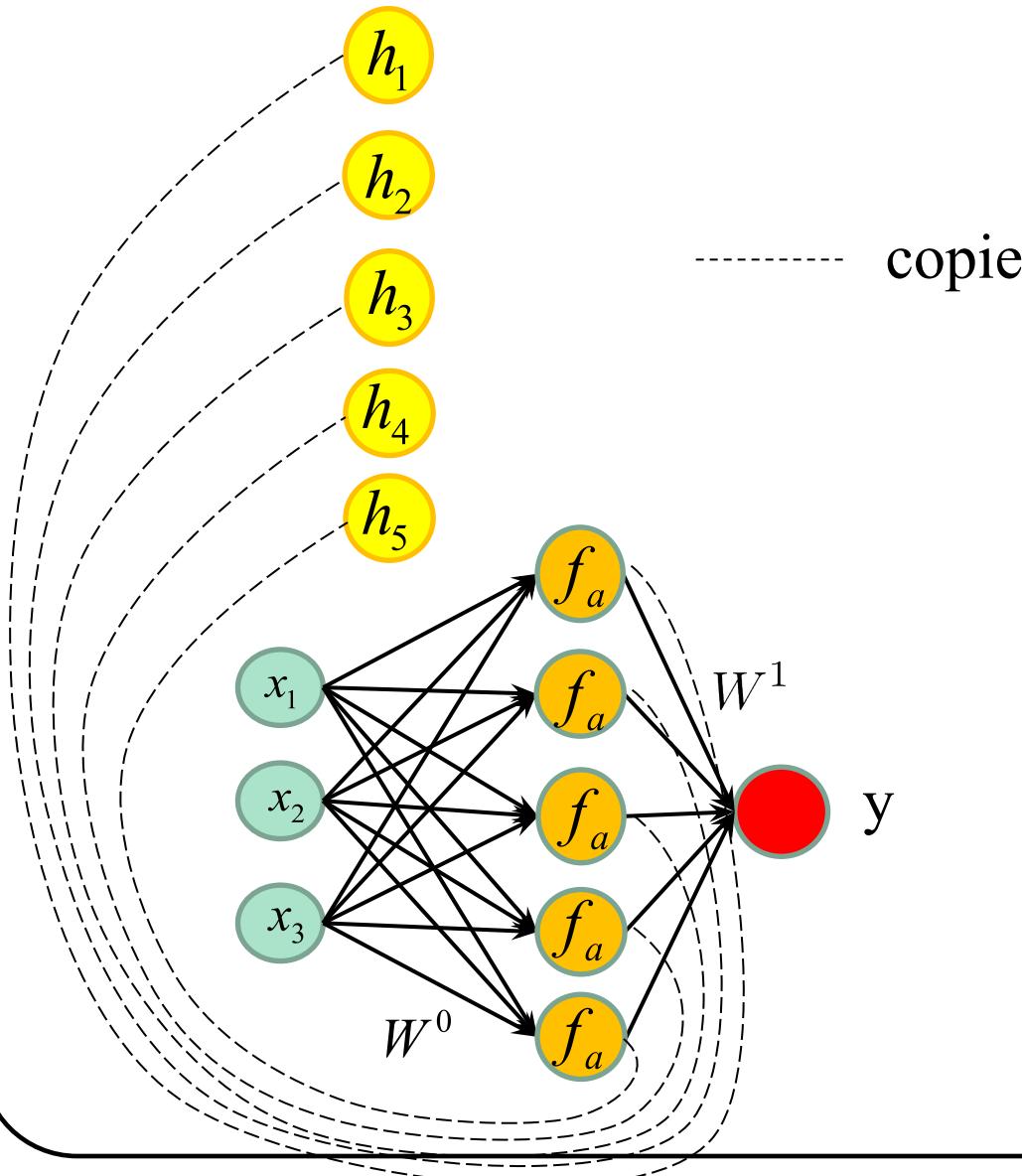


Ex.: Traduction Français-Anglais
N mots => M mots

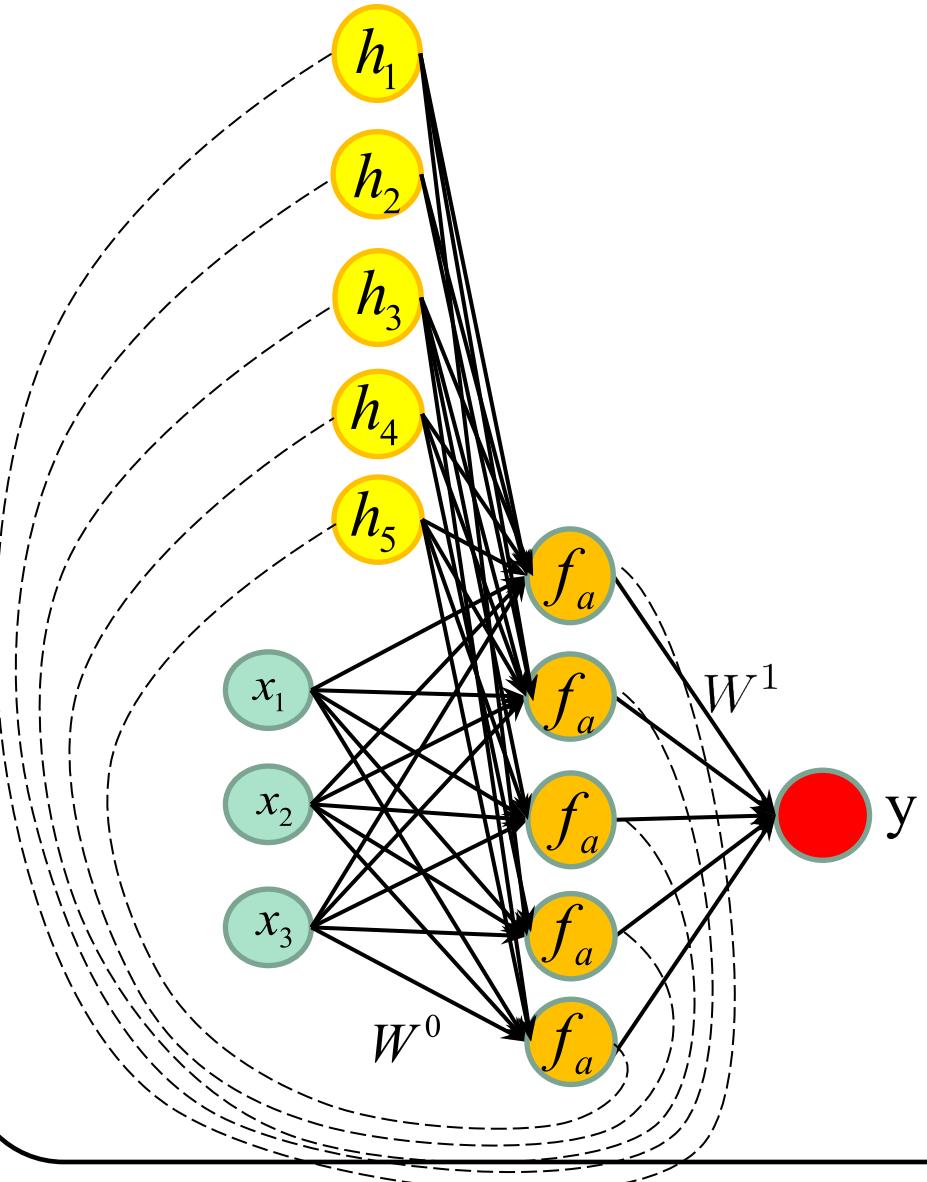
Réseau récurrent : la sortie des neurones est réinjectée dans leur entrée



Réseau récurrent : la sortie des neurones est réinjectée dans leur entrée

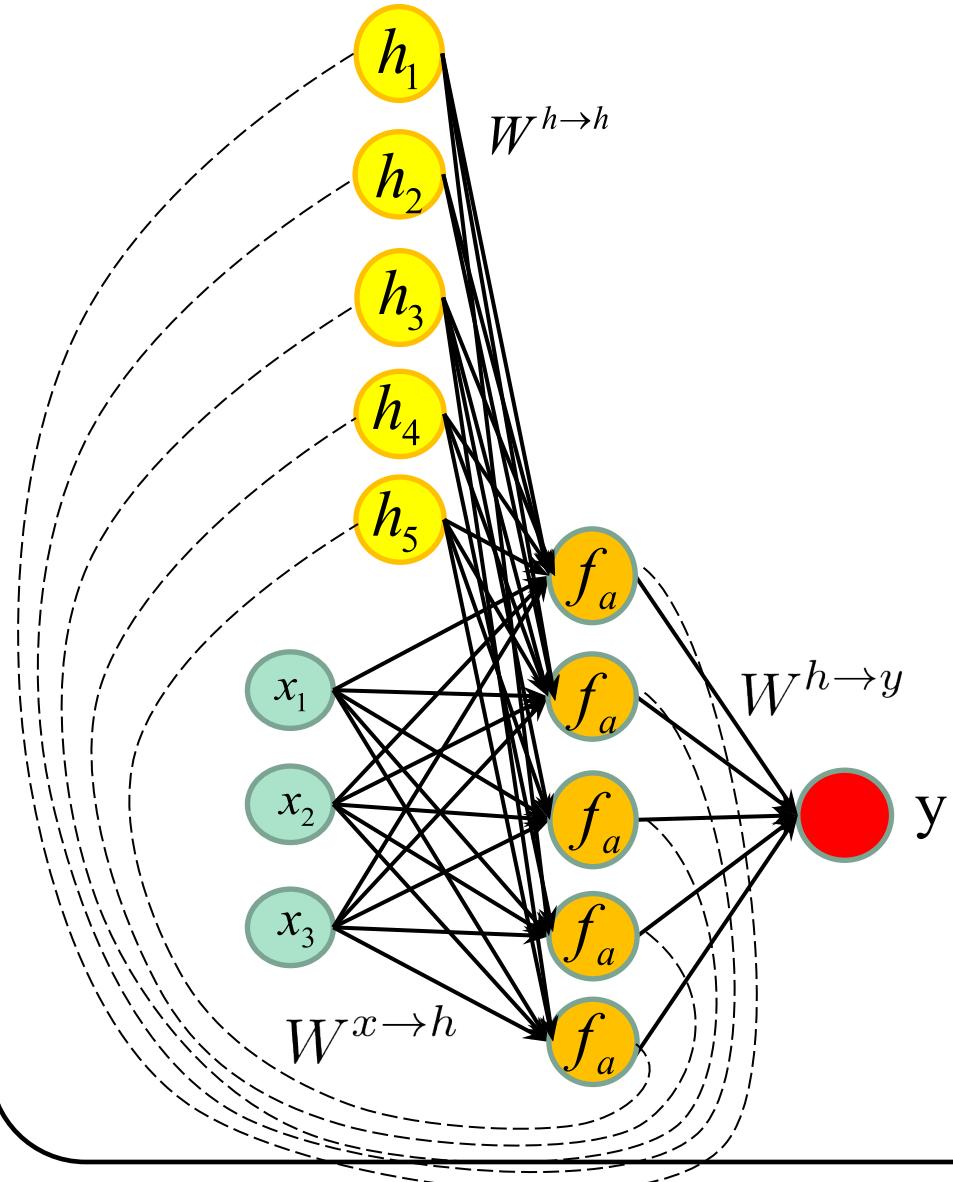


Réseau récurrent : la sortie des neurones est réinjectée dans leur entrée



Ici, au lieu d'avoir 3 entrées,
chaque neurone a **3+5=8** entrées.

Réseau récurrent : la sortie des neurones est réinjectée dans leur entrée



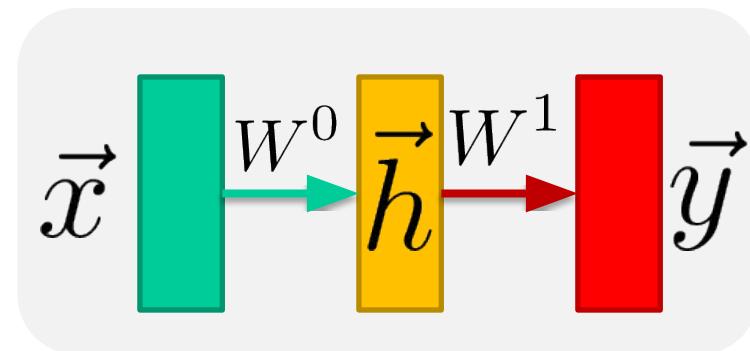
$$y(\vec{x}) = W^{h \rightarrow y} f_a(W^{x \rightarrow h} \vec{x} + W^{h \rightarrow h} \vec{h})$$



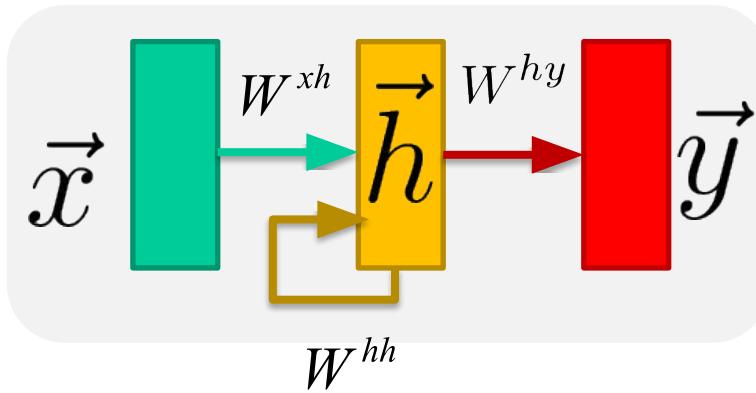
$$\vec{h} = f_a(W^{x \rightarrow h} \vec{x} + W^{h \rightarrow h} \vec{h})$$

$$y(\vec{x}) = W^{h \rightarrow y} \vec{h}$$

Illustration simplifiée

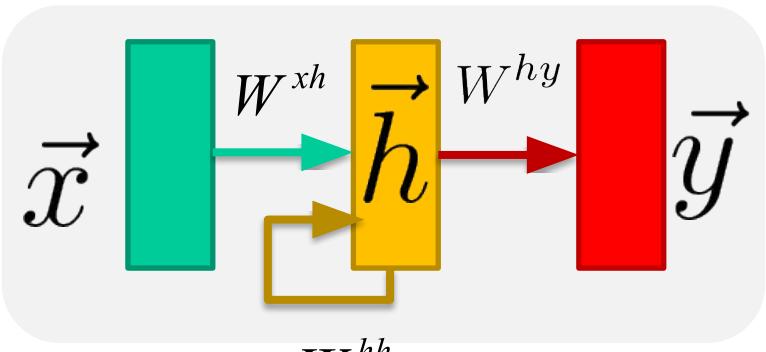


RN de base



RN récurrent

Dans le cas général avec K sorties (régression)



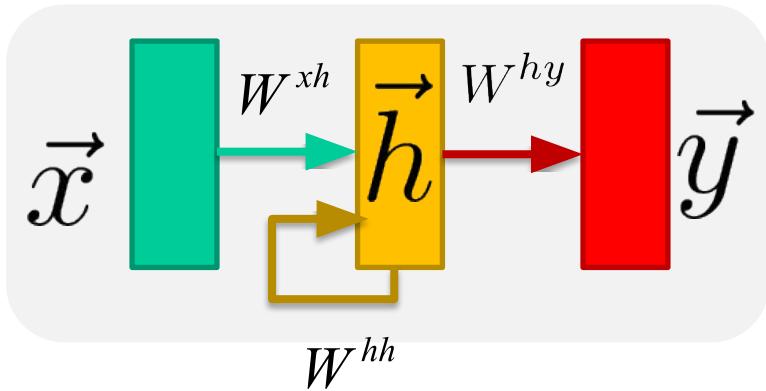
$$y(\vec{x}) = W^{hy} f_a(W^{xh} \vec{x} + W^{hh} \vec{h})$$



$$\vec{h} = f_a(W^{xh} \vec{x} + W^{hh} \vec{h})$$

$$\vec{y}(\vec{x}) = W^{hy} \vec{h}$$

Dans le cas général avec K sorties (classification)



$$y(\vec{x}) = W^{hy} f_a(W^{xh} \vec{x} + W^{hh} \vec{h})$$



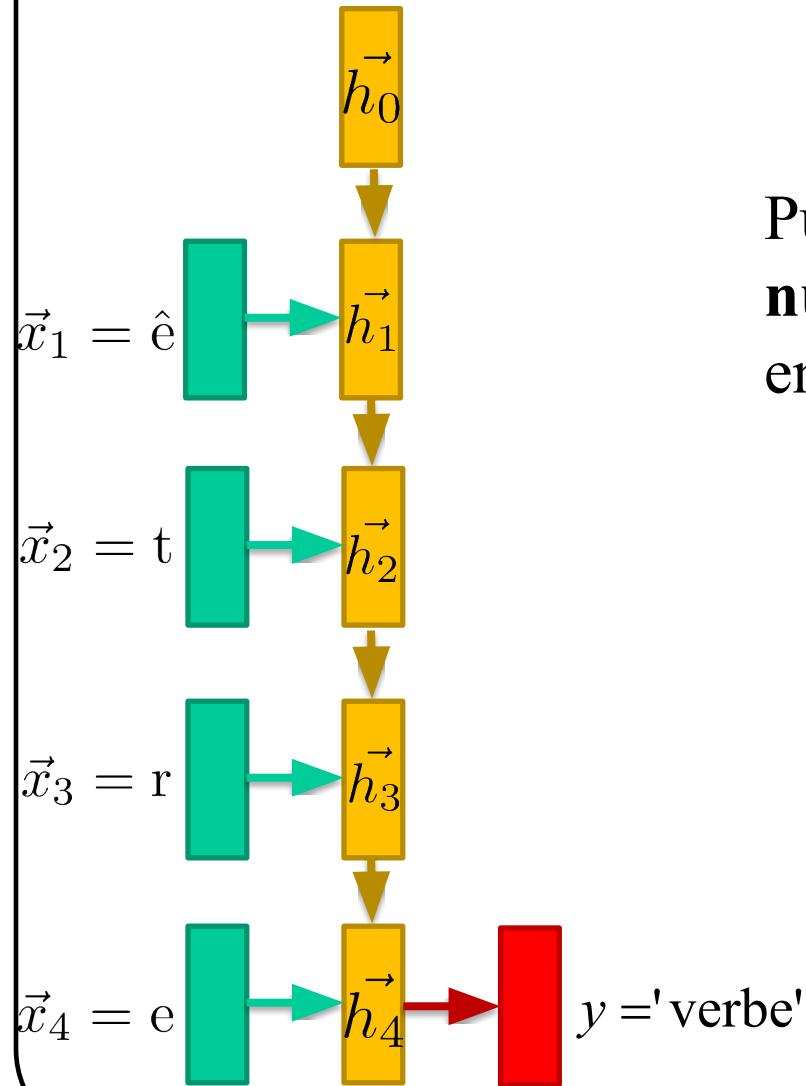
$$\vec{h} = f_a(W^{xh} \vec{x} + W^{hh} \vec{h})$$

$$\hat{y} = W^{hy} \vec{h}$$

$$\vec{y}(\vec{x}) = \text{softmax}(\hat{y})$$

Exemple pour N entrées et 1 sortie:

Analyse grammaticale (classification) : (ê.t.r.e)=>'verbe'

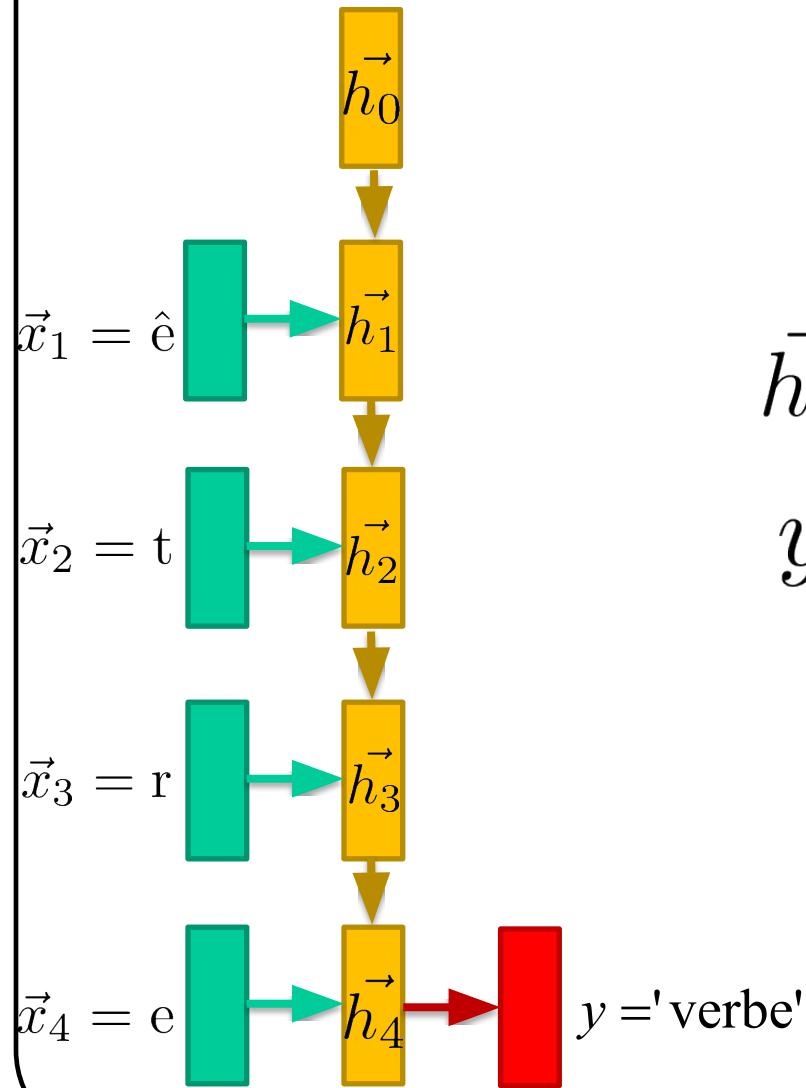


Puisque \vec{x} , \vec{h} et y doivent être des **variables numériques** on utilise généralement un encodage de type « *one hot* ».

$$\left. \begin{array}{l} 'a' = [1, 0, 0, \dots, 0] \\ 'b' = [0, 1, 0, \dots, 0] \\ 'c' = [0, 0, 1, \dots, 0] \\ \dots \\ 'verbe' = [1, 0, 0, \dots, 0] \\ 'nom' = [0, 1, 0, \dots, 0] \\ 'adjectif' = [0, 0, 1, \dots, 0] \end{array} \right\} \in R^{256}$$
$$\left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \in R^M$$

Exemple pour N entrées et 1 sortie:

Analyse grammaticale (classification) : (ê.t.r.e)=>'verbe'

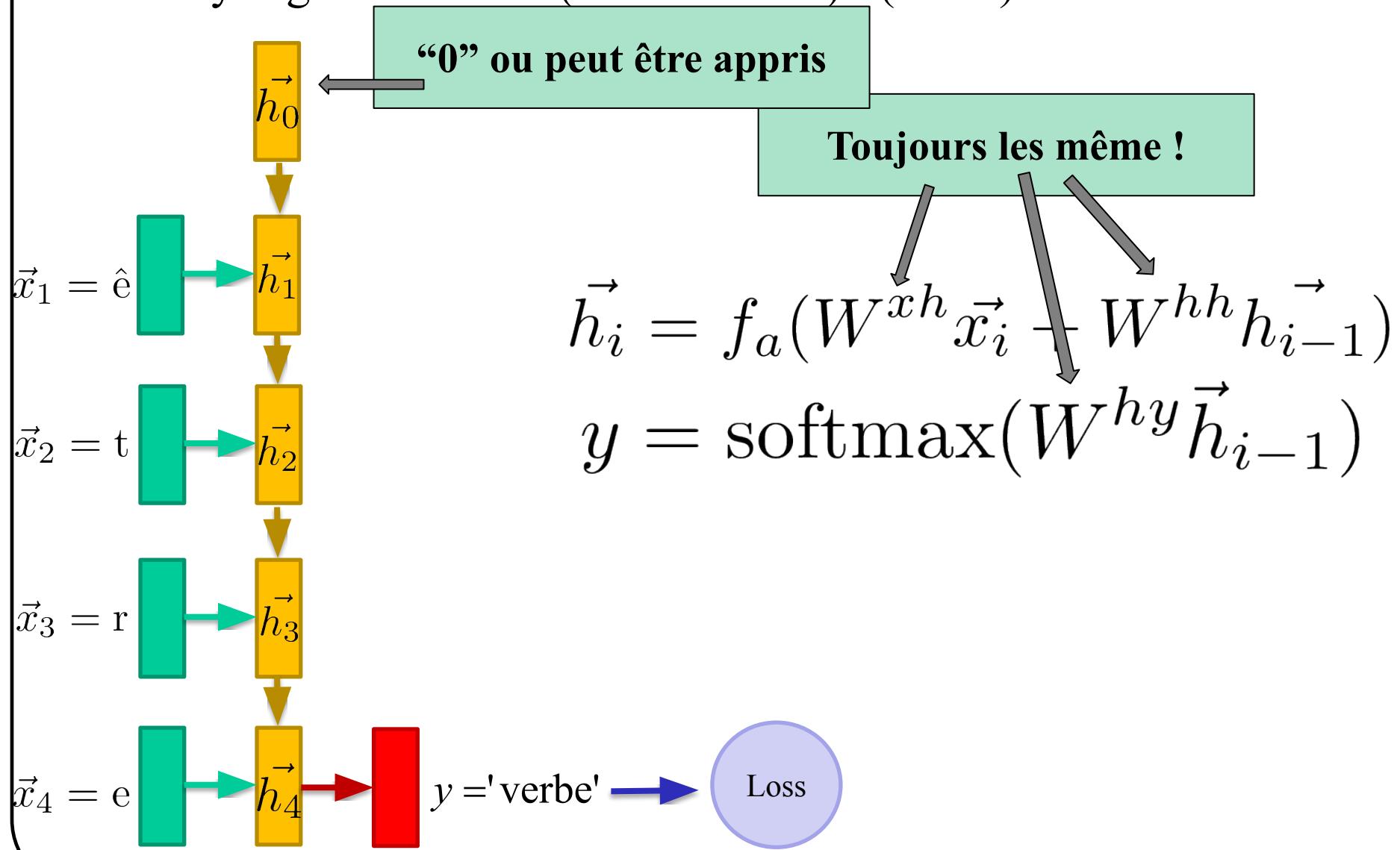


$$\vec{h}_i = f_a(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

$$y = \text{softmax}(W^{hy}\vec{h}_{i-1})$$

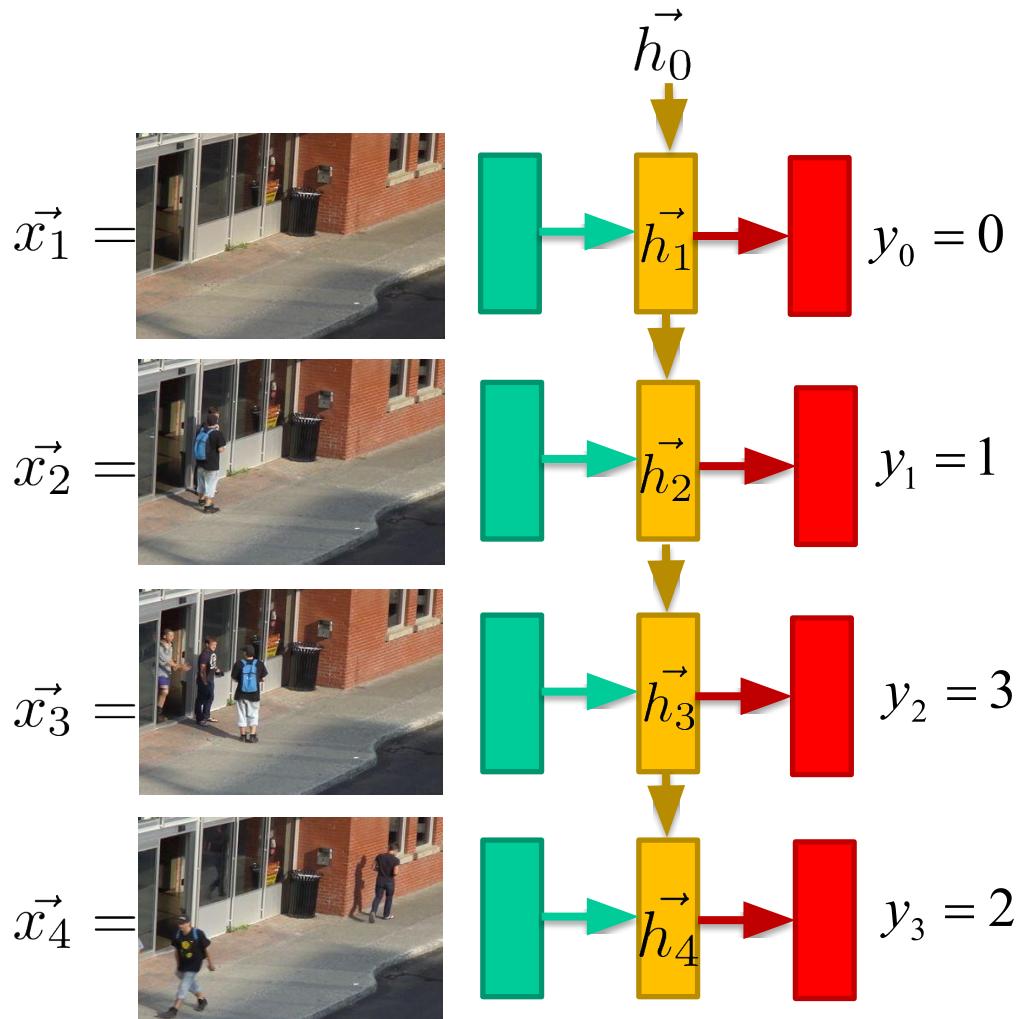
Exemple pour N entrées et 1 sortie:

Analyse grammaticale (classification) : (ê.t.r.e)=>'verbe'



Même idée pour N entrées et N sorties:

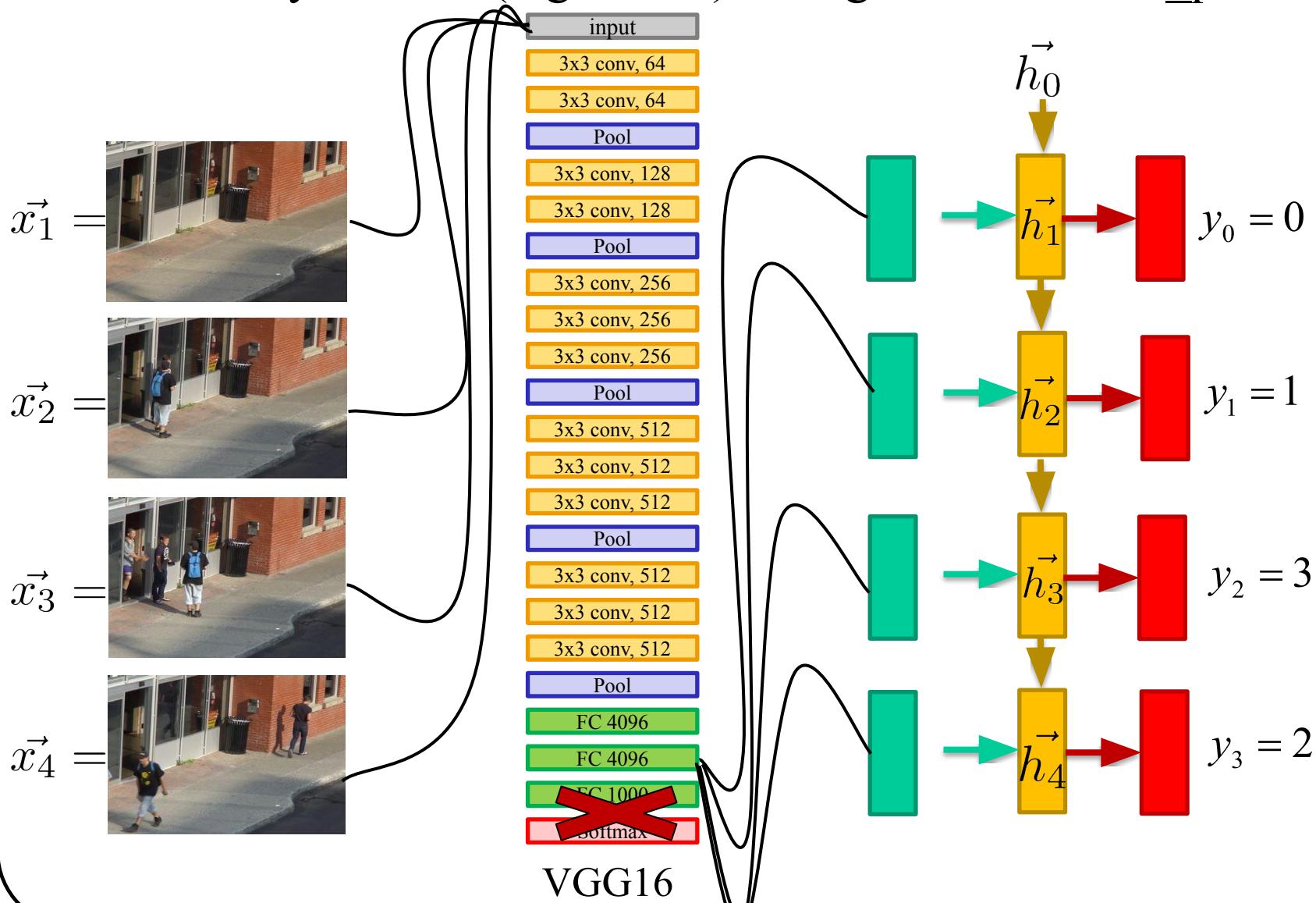
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



$$\begin{aligned}\vec{h}_i &= f_a(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1}) \\ y_i &= W^{hy} \vec{h}_i\end{aligned}$$

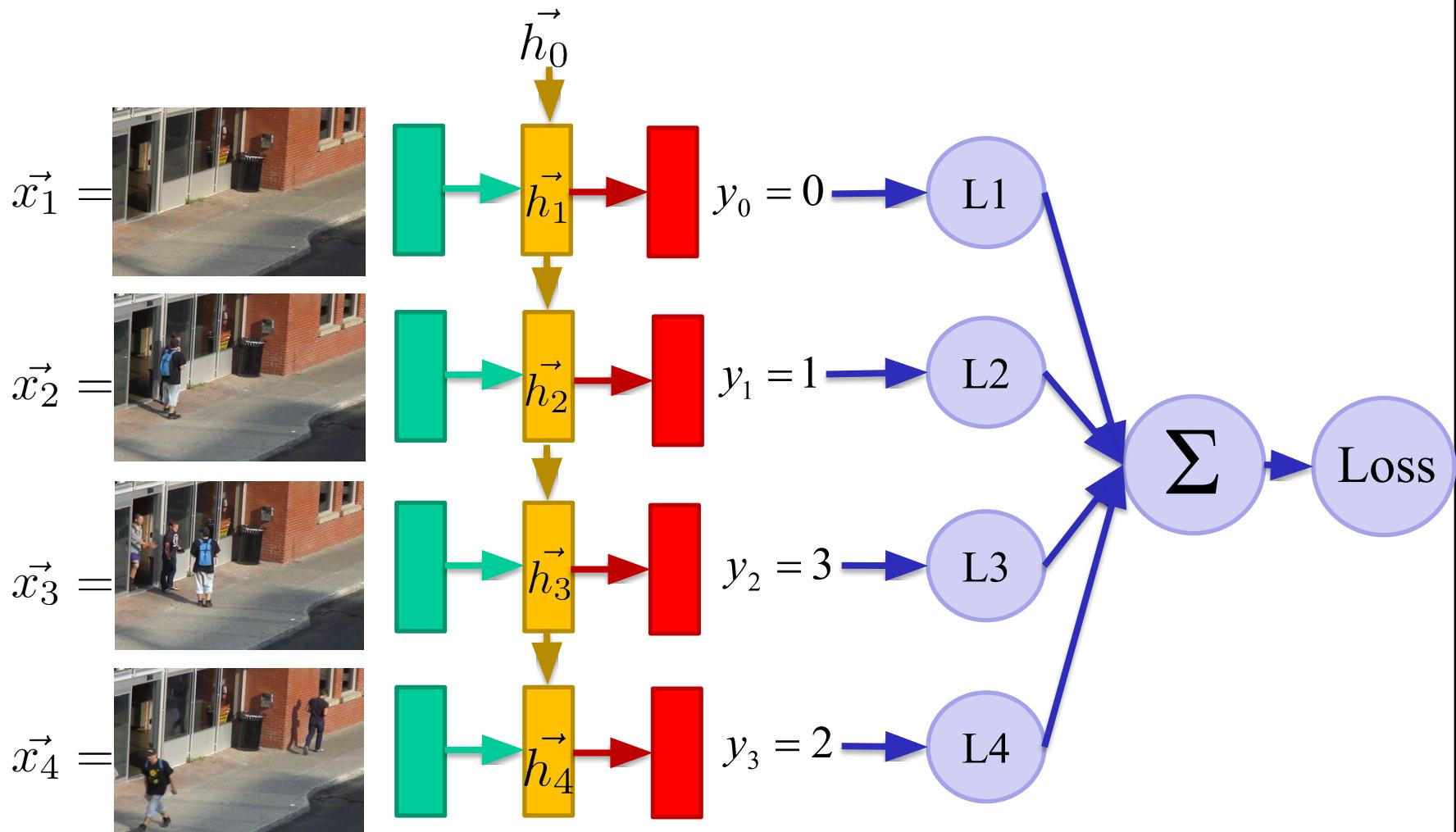
Même idée pour N entrées et N sorties:

ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



Même idée pour N entrées et N sorties:

ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet jouet :[a,e,m,s]

Représentation « one hot » jouet:

‘a’ = [1, 0, 0, 0]

‘e’ = [0, 1, 0, 0]

‘m’ = [0, 0, 1, 0]

‘s’ = [0, 0, 0, 1]

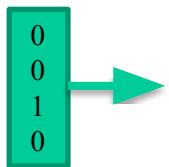
But : Entraîner un modèle à prédire les lettres du mot « **masse** ».

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

Entraîner un modèle à prédire les lettres du mot « **masse** ».

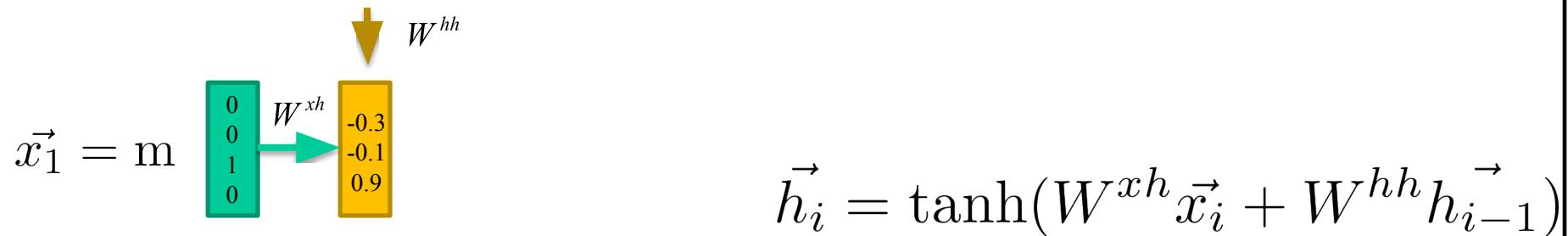
$$\vec{x}_1 = m$$



Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

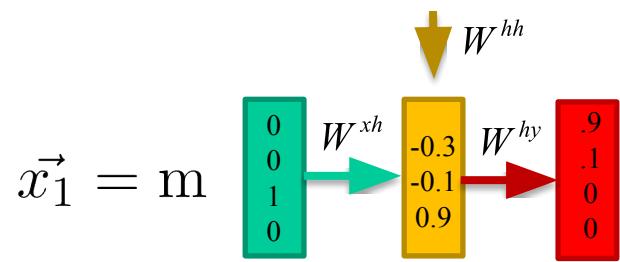
Entraîner un modèle à prédire les lettres du mot « **masse** ».



Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

Entraîner un modèle à prédire les lettres du mot « **masse** ».



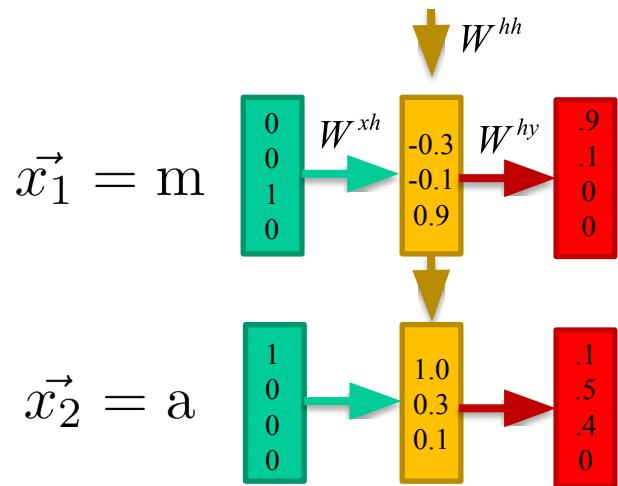
$$\vec{h}_i = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

$$\vec{y}(\vec{x}_i) = \text{softmax}(W^{hy}\vec{h}_i)$$

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

Entraîner un modèle à prédire les lettres du mot « **masse** ».



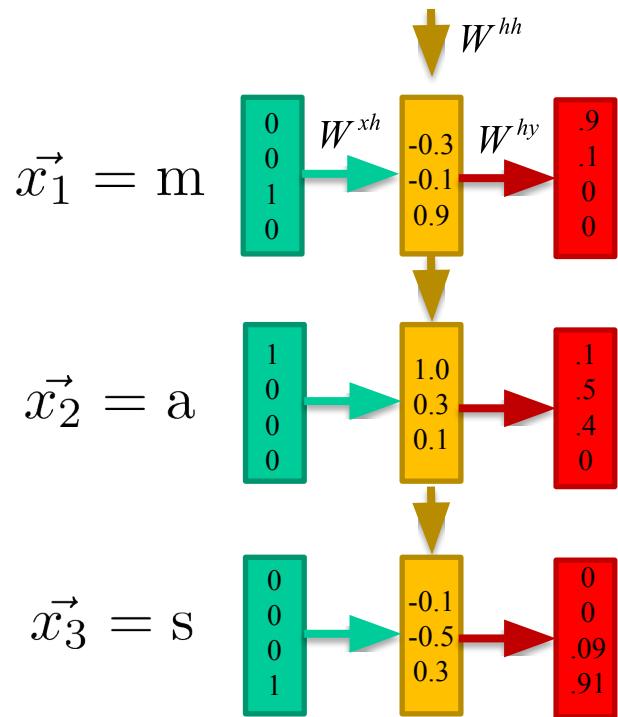
$$\vec{h}_i = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

$$\vec{y}(\vec{x}_i) = \text{softmax}(W^{hy}\vec{h}_i)$$

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

Entraîner un modèle à prédire les lettres du mot « **masse** ».

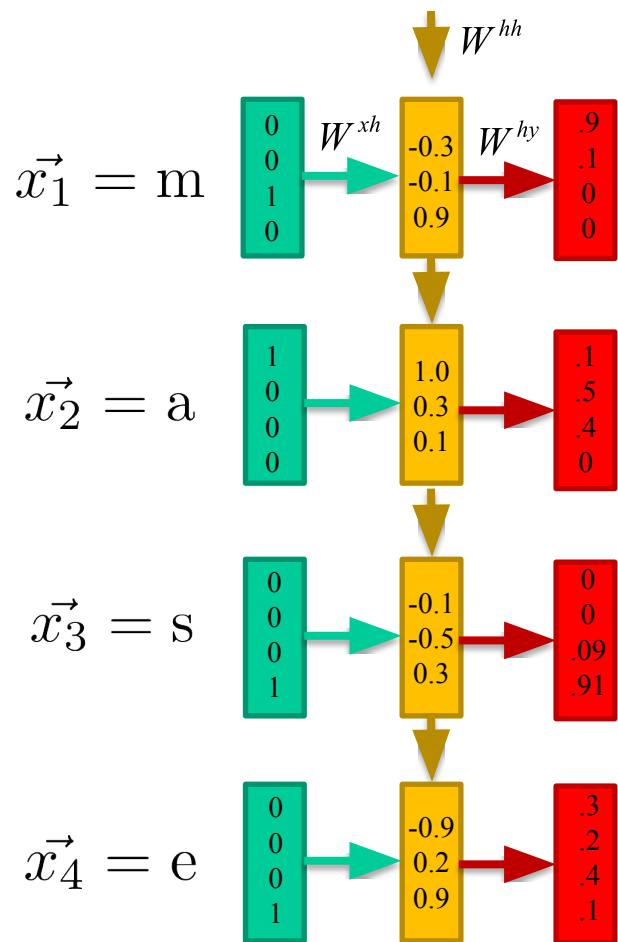


$$\vec{h}_i = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$
$$\vec{y}(\vec{x}_i) = \text{softmax}(W^{hy}\vec{h}_i)$$

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

Entraîner un modèle à prédire les lettres du mot « **masse** ».

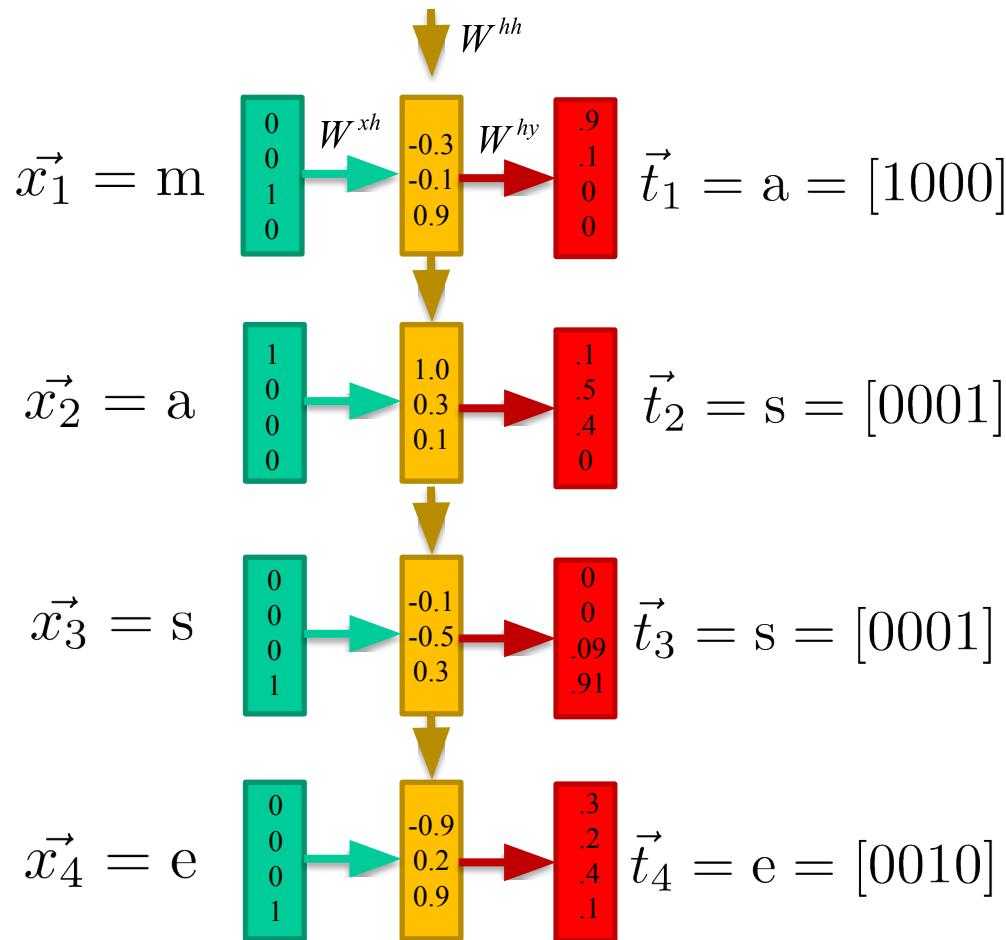


$$\vec{h}_i = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$
$$\vec{y}(\vec{x}_i) = \text{softmax}(W^{hy}\vec{h}_i)$$

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

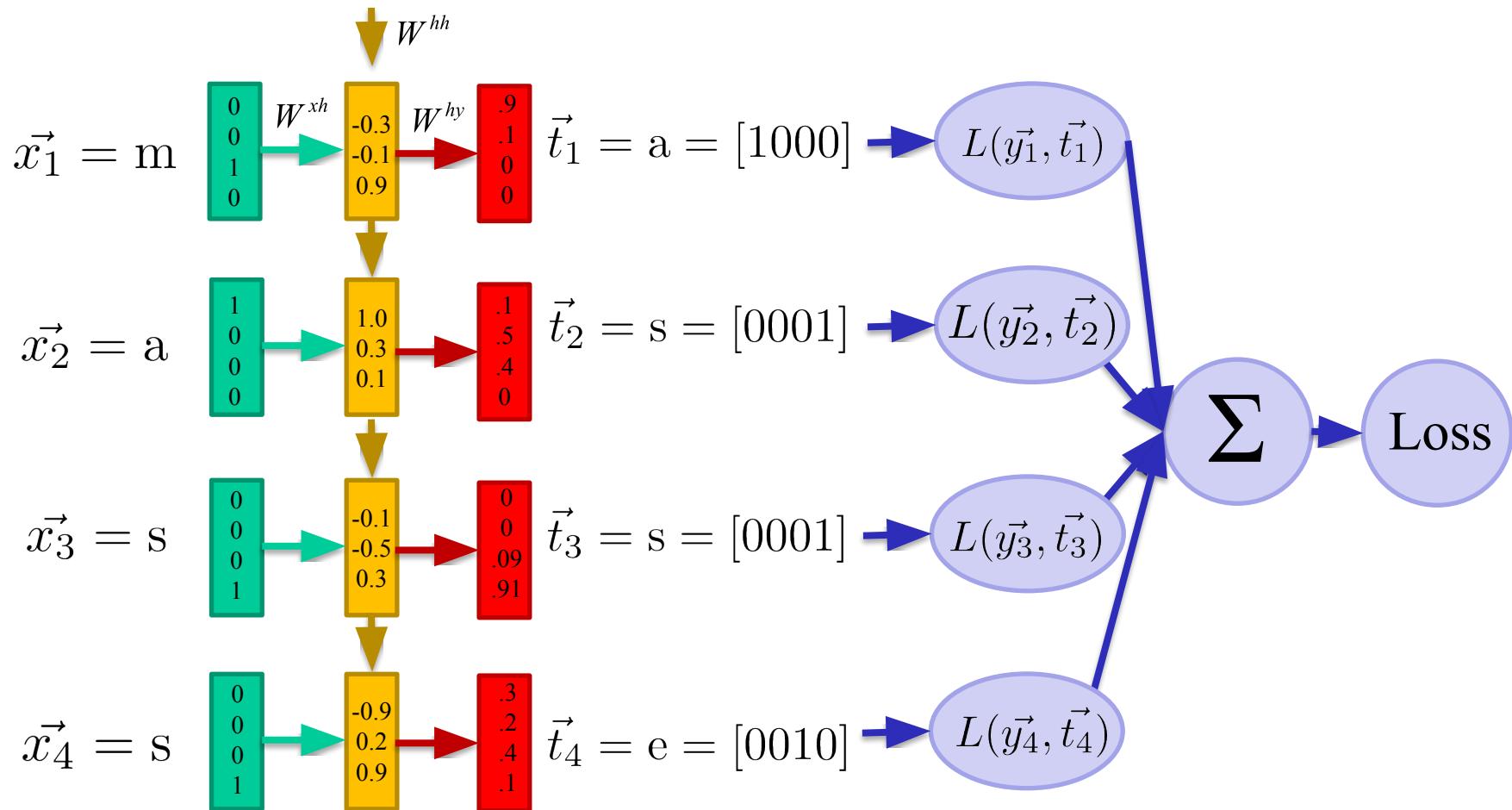
Entraîner un modèle à prédire les lettres du mot « **masse** ».



Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

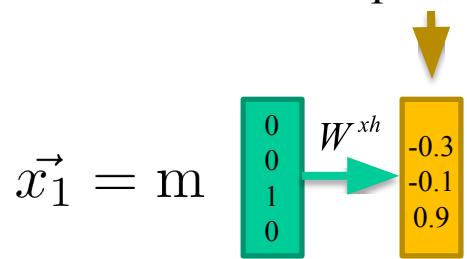
Entraîner un modèle à prédire les lettres du mot « **masse** ».



Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

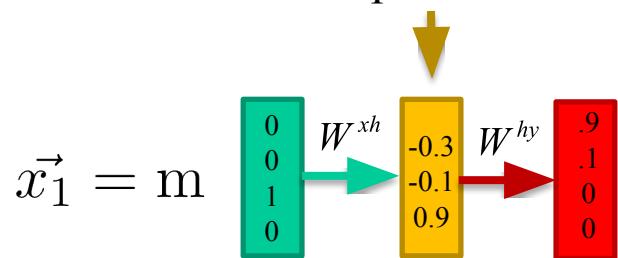


Étape 1 : Calcul de la couche cachée

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

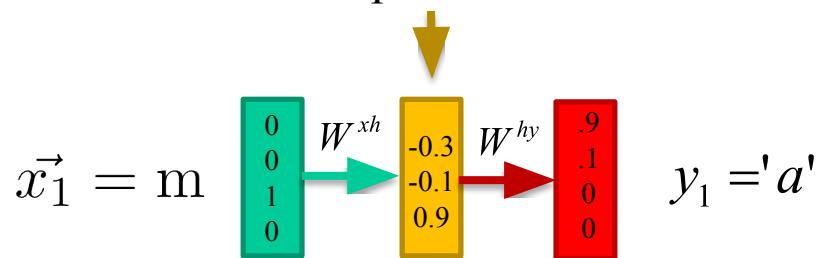


Étape 2 : Calcul de la sortie (softmax)

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

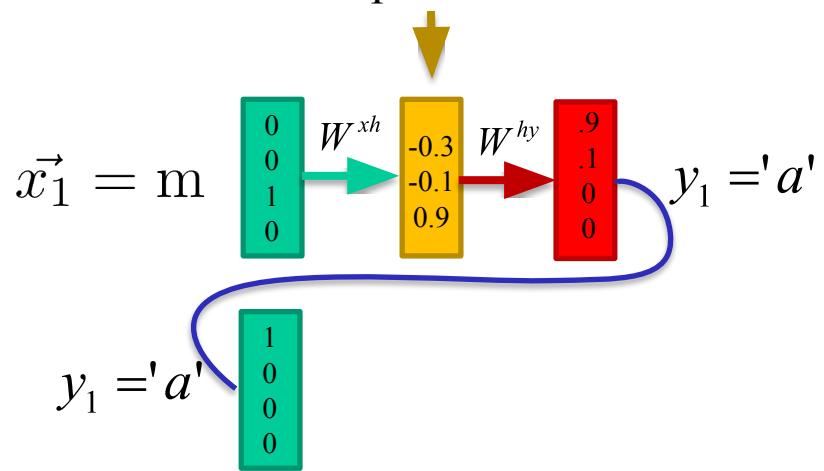


Étape 3 : Sélectionner le caractère le plus probable

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

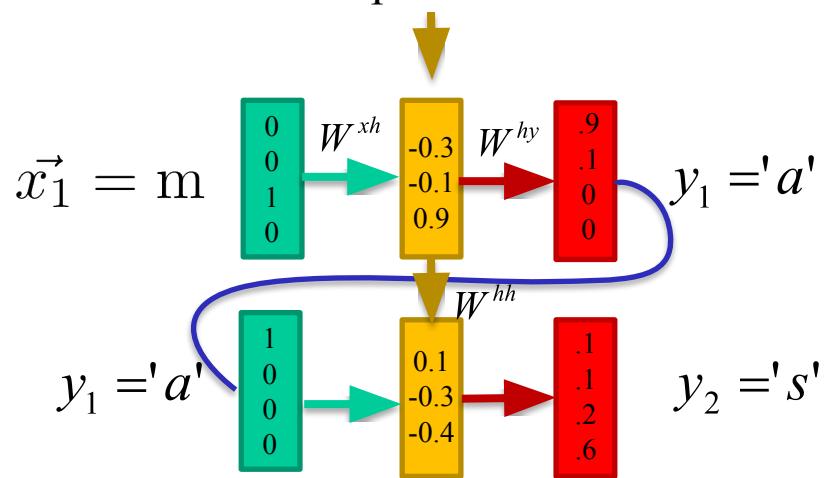


Étape 4 : Injecter le caractère prédict au début du réseau

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

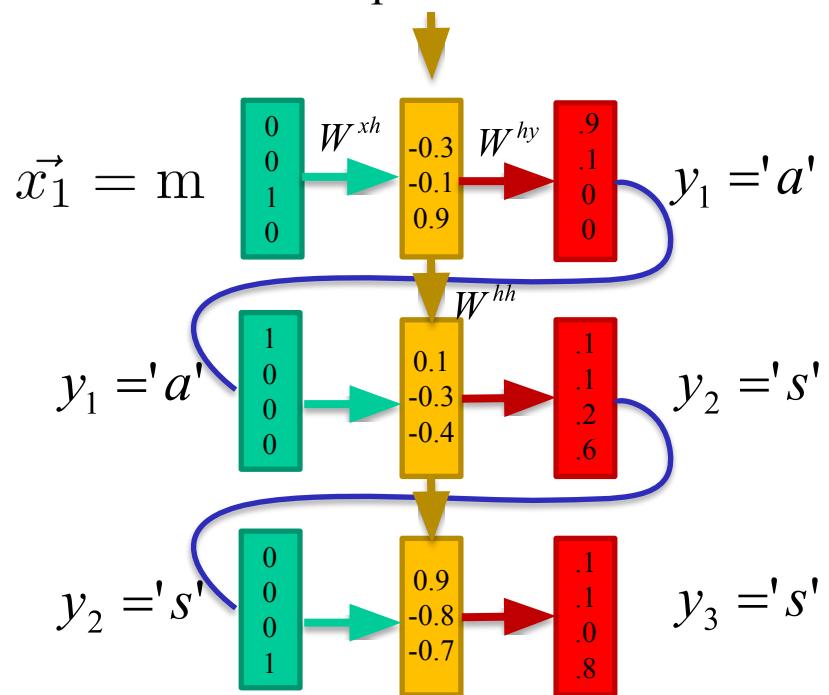


Et on recommence!

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

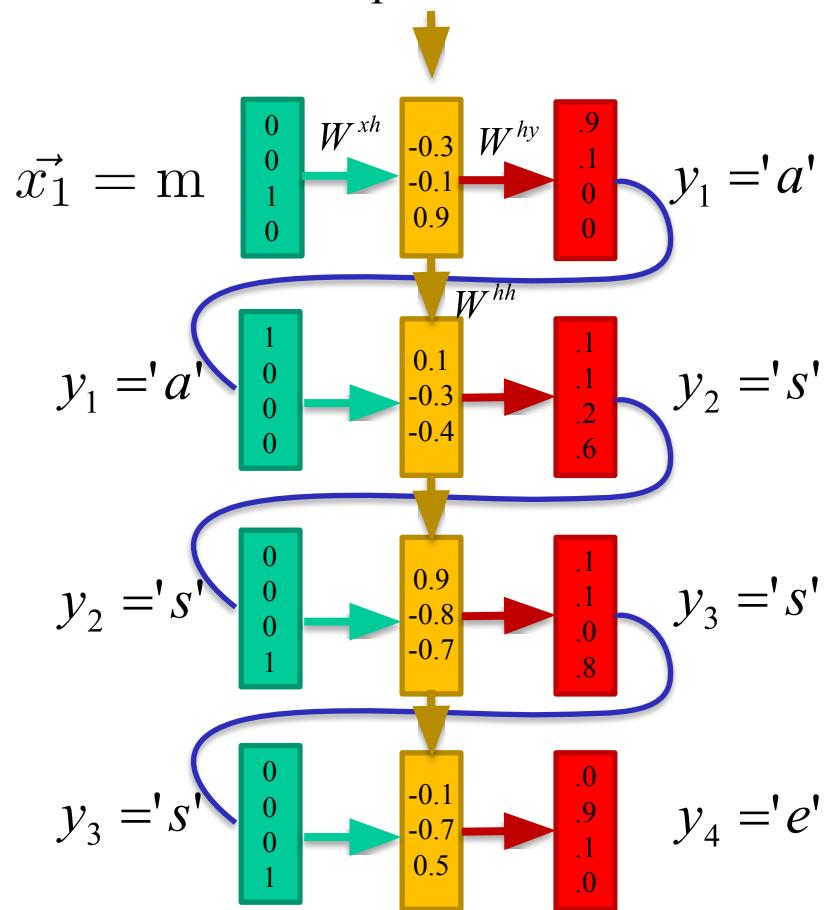
En test : prédire les lettres les unes après les autres



Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



Autre exemple: prédiction de caractères (modèle de langue)

Code python: “mini-char-RNN” de A. Karpathy

<https://gist.github.com/karpathy/d4dee566867f8291f086>

Un RNN en 112 lignes !

Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy

```
min-char-rnn.py
```

```
1 """
2 Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3 BSD License
4 """
5 import numpy as np
6
7 # data I/O
8 data = open('input.txt', 'r').read() # should be simple plain text file
9 chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
26
27 def lossFun(inputs, targets, hprev):
28 """
29 inputs,targets are both list of integers.
30 hprev is Hx1 array of initial hidden state
31 returns the loss, gradients on model parameters, and last hidden state
32 """
33 xs, hs, ys, ps = {}, {}, {}, {}
34 hs[-1] = np.copy(hprev)
35 loss = 0
36 # forward pass
37 for t in xrange(len(inputs)):
38     xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
39     xs[t][inputs[t]] = 1
```

$$\begin{aligned} 'a' &= [1, 0, 0, \dots, 0] \\ 'b' &= [0, 1, 0, \dots, 0] \\ 'c' &= [0, 0, 1, \dots, 0] \end{aligned} \quad \left. \right\} \in R^{256}$$

...

Autre exemple: prédiction de caractères (modèle de langue)

Code python: “mini-char-RNN” de A. Karpathy

<https://gist.github.com/karpathy/d4dee566867f8291f086>

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserved thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.

$$\begin{aligned} 'a' &= [1, 0, 0, \dots, 0] \\ 'b' &= [0, 1, 0, \dots, 0] \\ 'c' &= [0, 0, 1, \dots, 0] \end{aligned} \quad \left. \right\} \in R^{256}$$

...

Autre exemple: prédiction de caractères (modèle de langue)

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.

Autre exemple: prédiction de caractères (modèle de langue)

```
----  
iter 1816300, loss: 39.494453  
----  
nnes troi la nulits avovez son des préchâté, ne viche loi. Et serrité, ou présermoisse sent soclaine,  
cocraine, de la mas était pas. Devant lui quelque rumisent du cria entrois cepthib. Le _pouv  
----  
iter 1816400, loss: 39.437285  
----  
s. Qu'est-c'était bâciante Clemigant du passée avez-logue une salée:  
--Ains, honteatiées d'arrapa, à, emporle, Cosettes doncait. Marius cet hombre. Il a hume  
grêts on éline, ce sans tous l  
----  
iter 1816500, loss: 39.626851  
----  
côtée, et même des sordia: Crédit: A. Karpathy, CS231  
--Il sunques nous exation de toleva entré, comme-hes  
rivorme de rassieu. On a-Autant de bêtes, avernorbre que pierre dejoil, l'agraître. Disant dose de  
l'édet■  
----
```

Entraînement sur les cinq tomes de “Les misérables” provenant du *Gutenberg Project*
(<https://www.gutenberg.org/>)

Autre exemple: prédiction de caractères (modèle de langue)

l'eccure, à la détreste
probée.

«Elle qu'il senconches, plutige. Envreur de minetiesait soulèretominante? l'ontatilletier si moin, ell
e hautôt sa trieur,
ou serace, tanne, les inquiqueur

iter 994600, loss: 41.919709

avait à retrí dideux la sailtirssique
plus autré pars une soir finions M...- Jes marturiteme. Le chemtite un camme tembres! quand avait Litat
ions au moursues là vien de diclit-ilur et--- inguses, c

iter 994700, loss: 41.791706

te, cielle, s'un dune autresses. Il surclait. Et par sur le saute, son êtres
come; faiement le bacevielle un
vandait la
pranconsenteu
de la croboti:

Entraînement sur Madame Bovary provenant du *Gutenberg Project*
(<https://www.gutenberg.org/>)

Autre exemple: prédiction de caractères (modèle de langue)

Texte généré une fois le modèle entraîné

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

VIOLA:

Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

Autre exemple: prédiction de caractères (modèle de langue)

Entraînement sur le code source de Linux en C++

Texte généré une fois le modèle entraîné

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffff8) & 0x000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &offset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>

#define REG_PG      vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)      (func)

#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs() arch_get_unaligned_child()
#define access_rw(TST) asm volatile("movd %esp, %0, %3" :: "r" (0)); \
    if (_type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
                                              pC[1]);

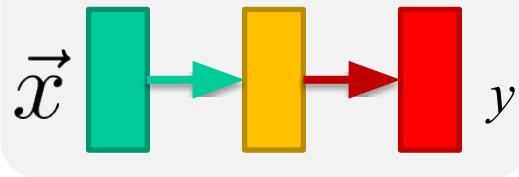
static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
                (unsigned long)-1->lr_full, low);
}
}
```

Différentes configurations pour différentes applications

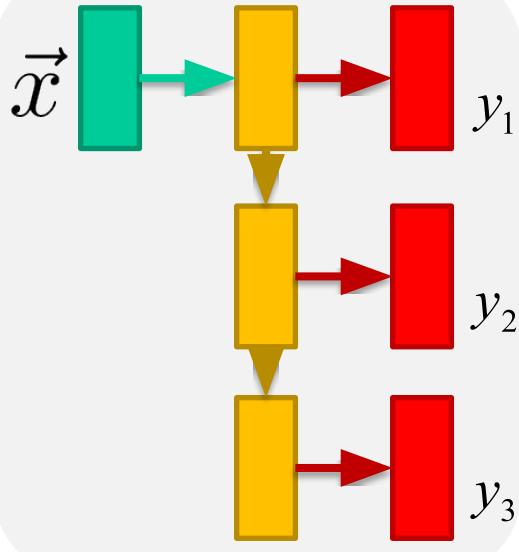
M entrées et N sorties

1 entrée et N sorties

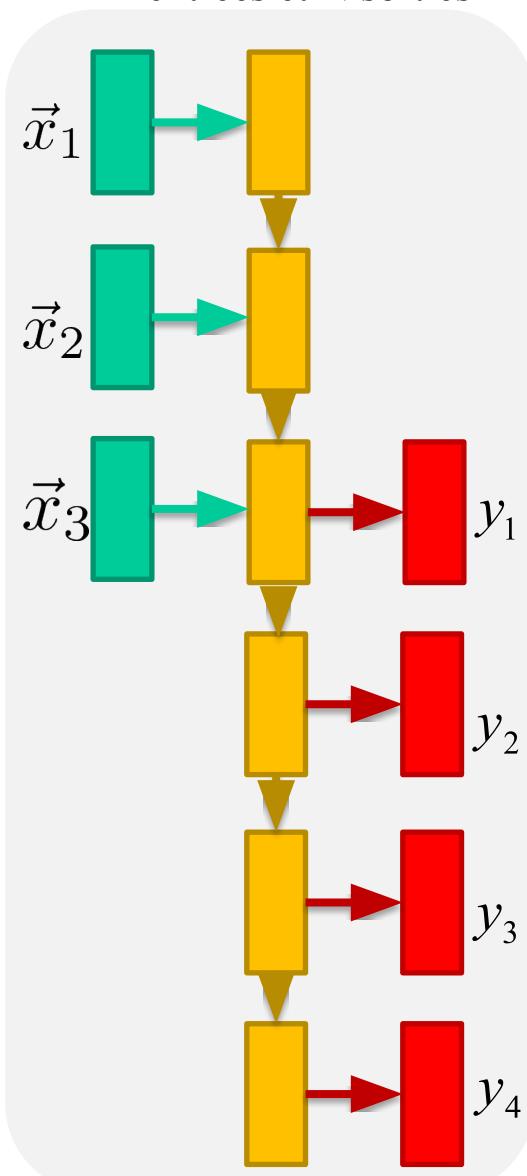
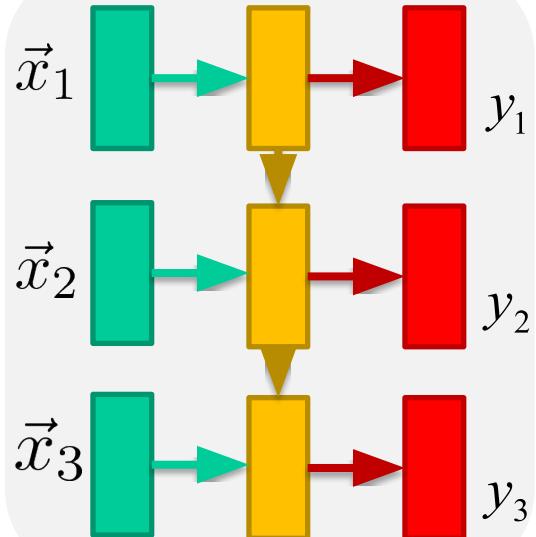
1 entrée et 1 sortie



1 entrée et N sorties



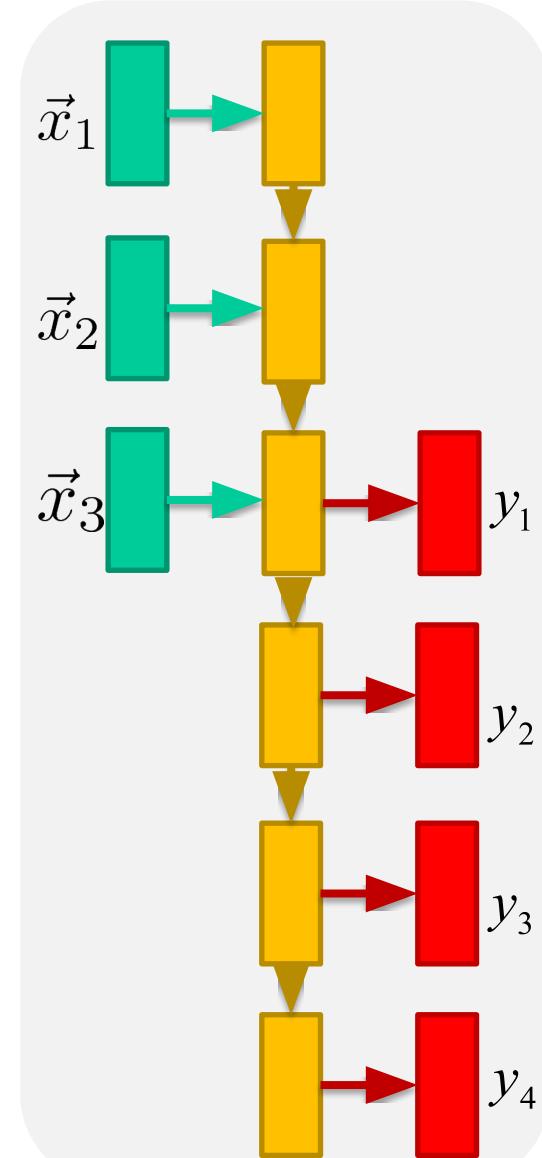
N entrées et N sorties



Ex.: Traduction Français-Anglais
N mots => M mots

Différentes configurations pour différentes applications

M entrées et N sorties



Ex.: Traduction Français-Anglais
N mots => M mots

Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]

Pas le même nombre d’entrées et de sorties !

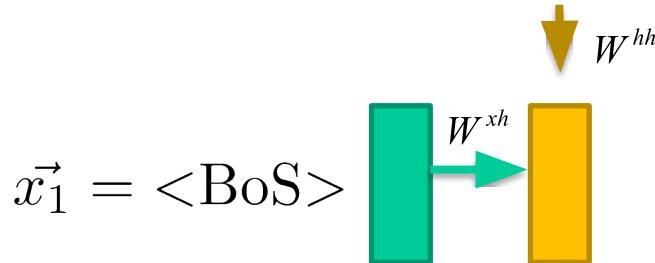
Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]



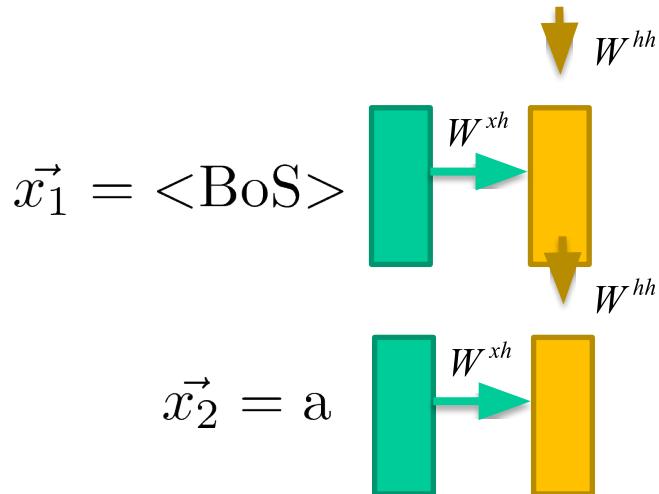
Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]



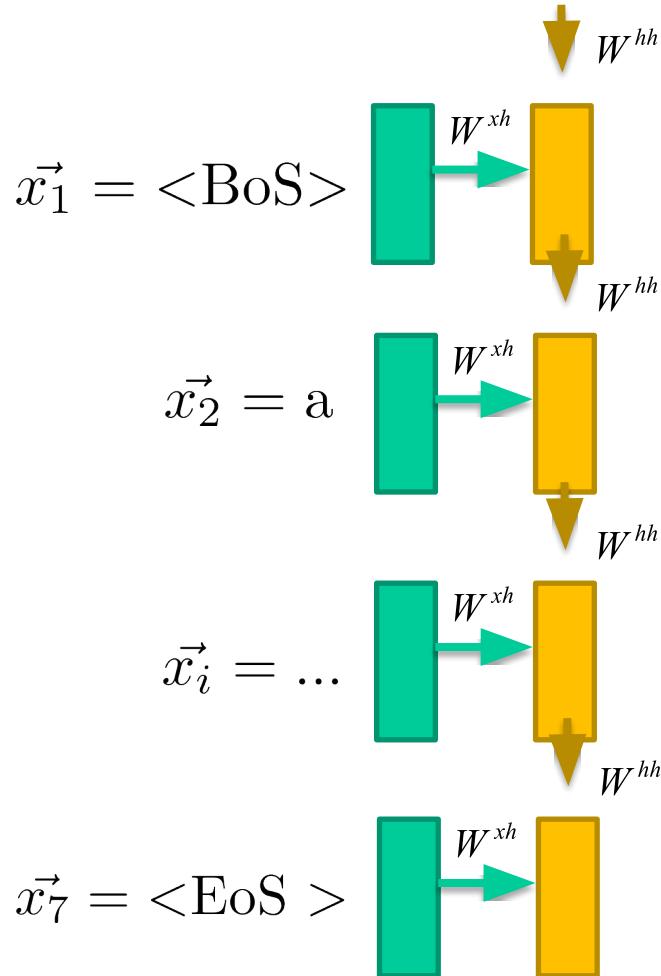
Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]



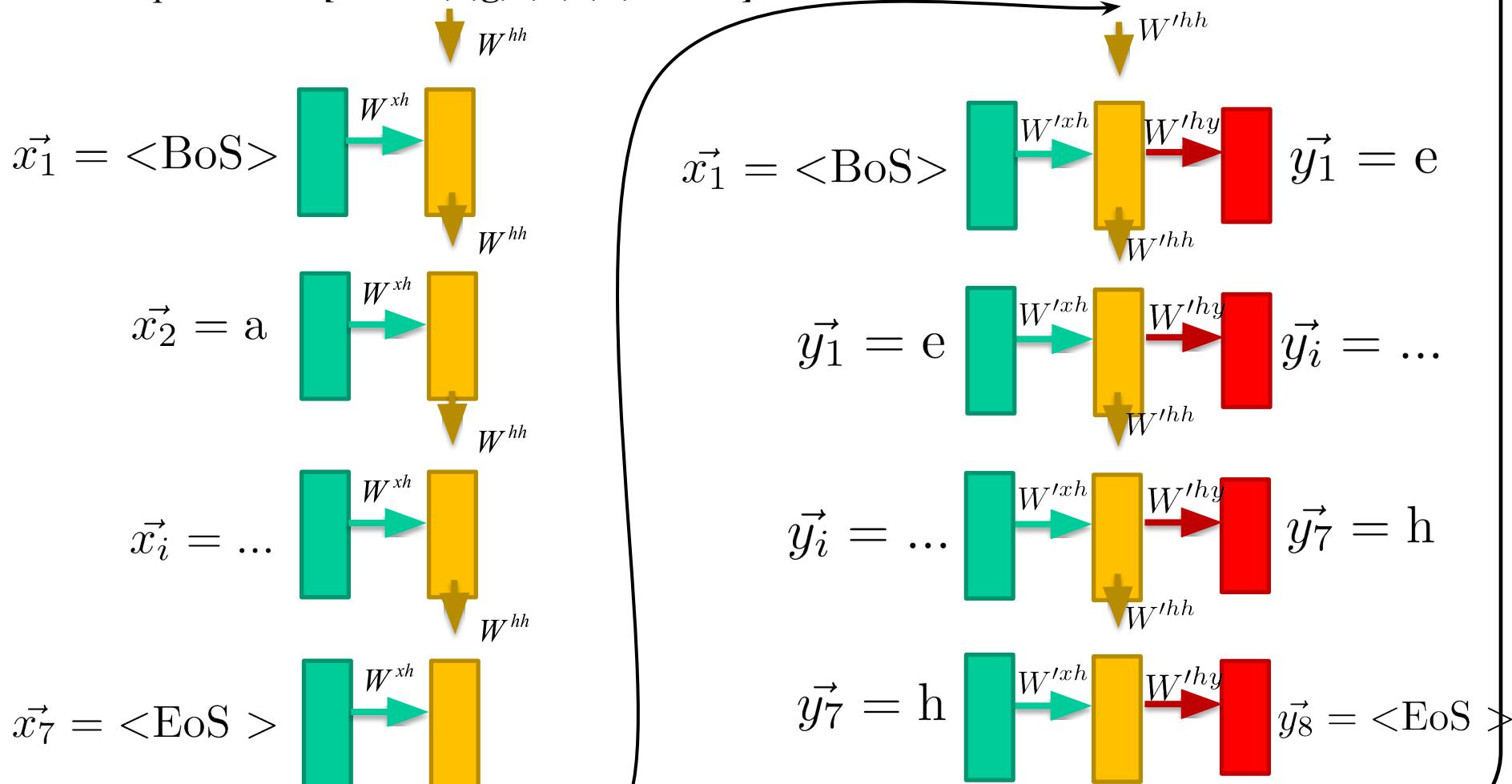
Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]



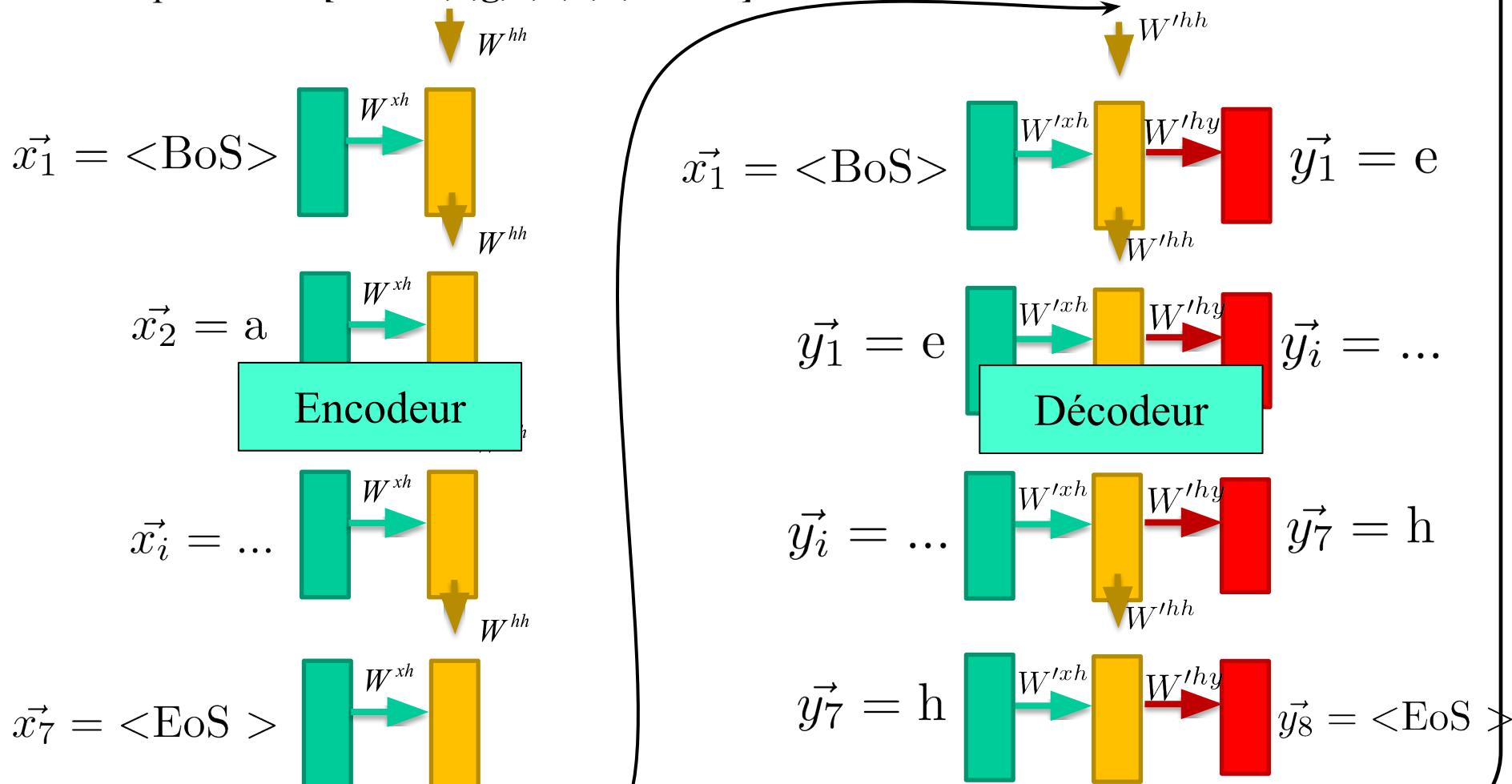
Autre exemple: traduction

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Traduire ‘assez’ -> ‘enough’

Alphabet fr :[<BoS>,a,e,s,z,<EoS>]

Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]

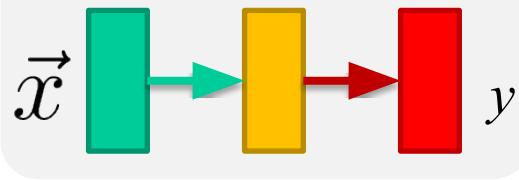


Différentes configurations pour différentes applications

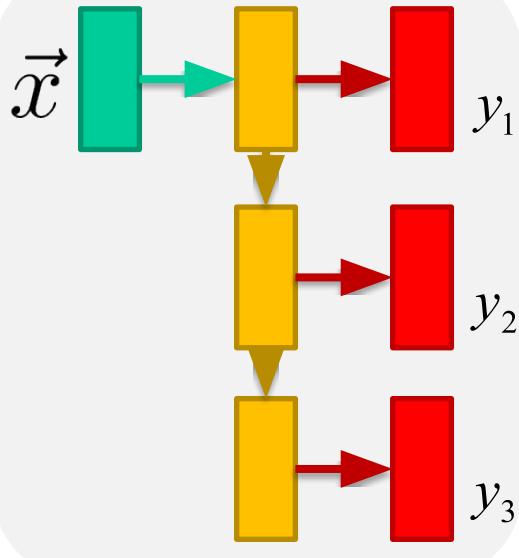
M entrées et N sorties

1 entrée et N sorties

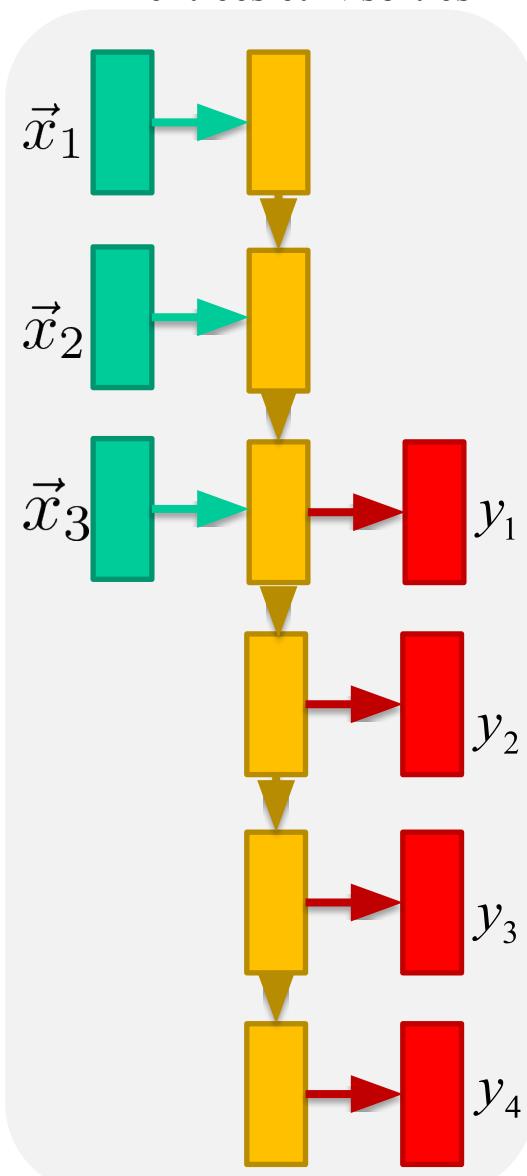
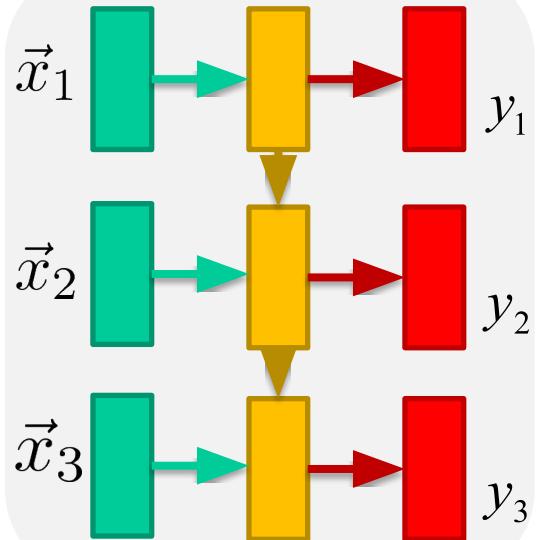
1 entrée et 1 sortie



1 entrée et N sorties



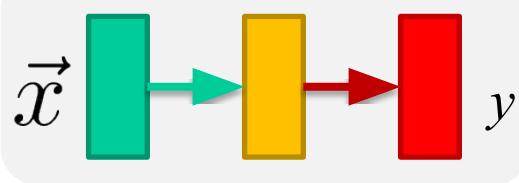
N entrées et N sorties



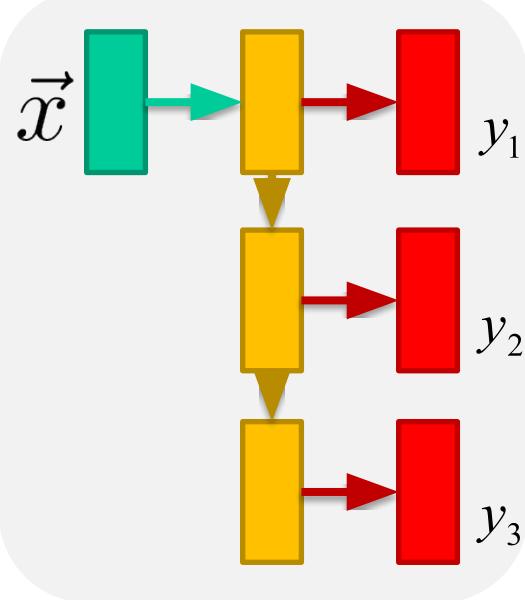
Ex.: Traduction Français-Anglais
N mots => M mots

Différentes configurations pour différentes applications

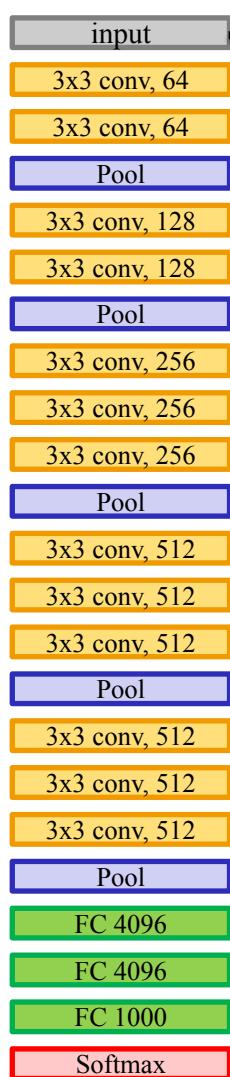
1 entrée et 1 sortie



1 entrée et N sorties

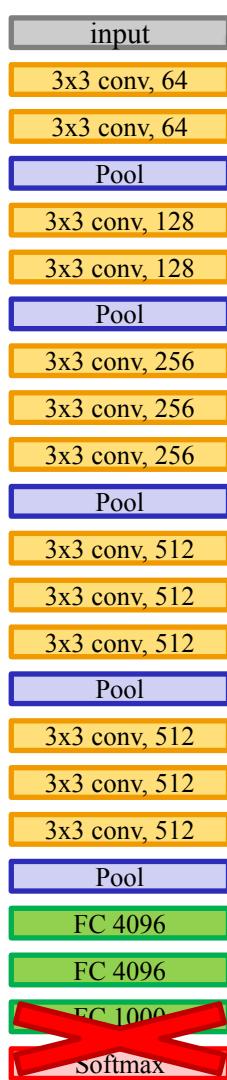


Captioning



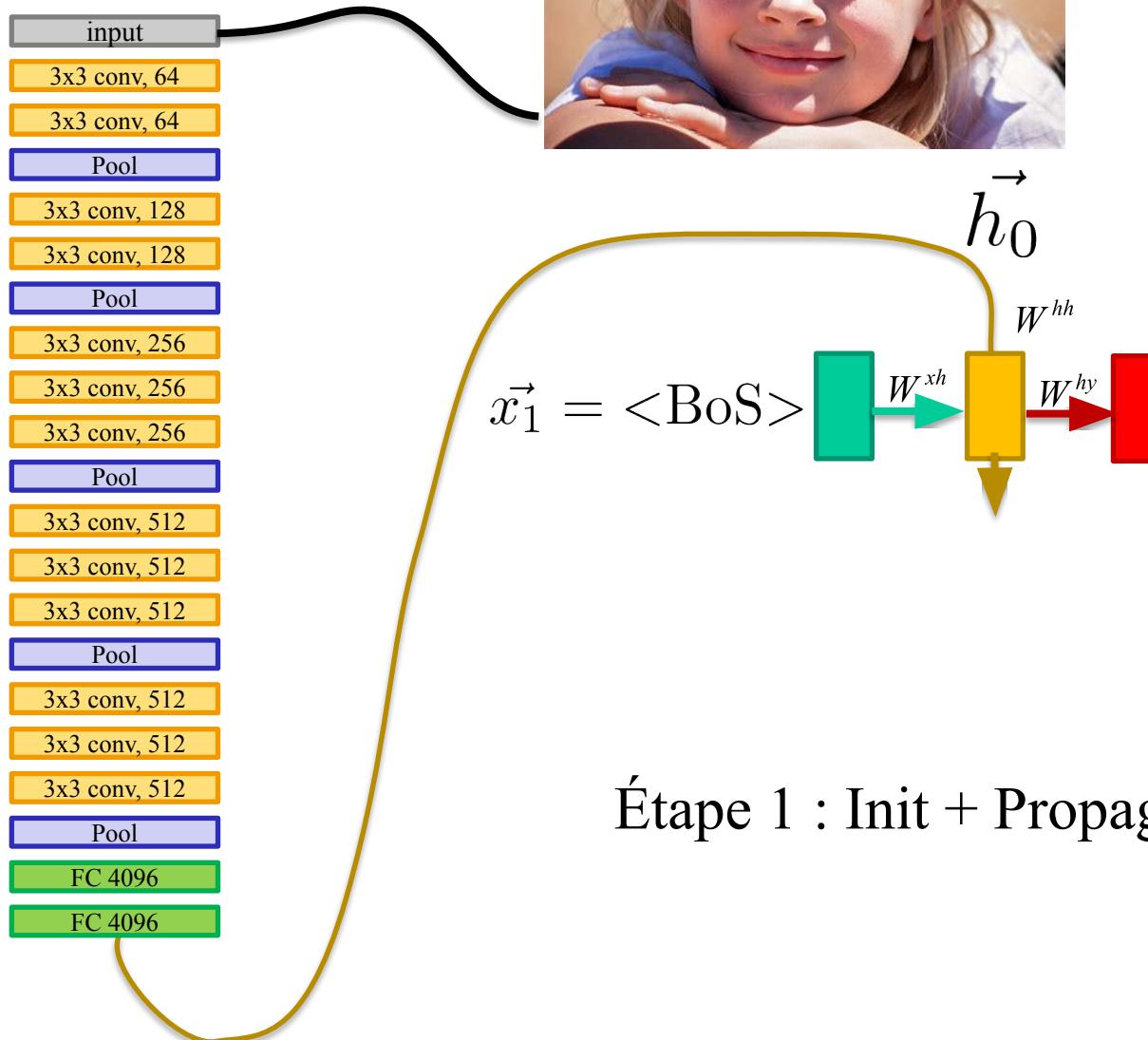
Pré-entraîné sur *ImageNet*

Captioning

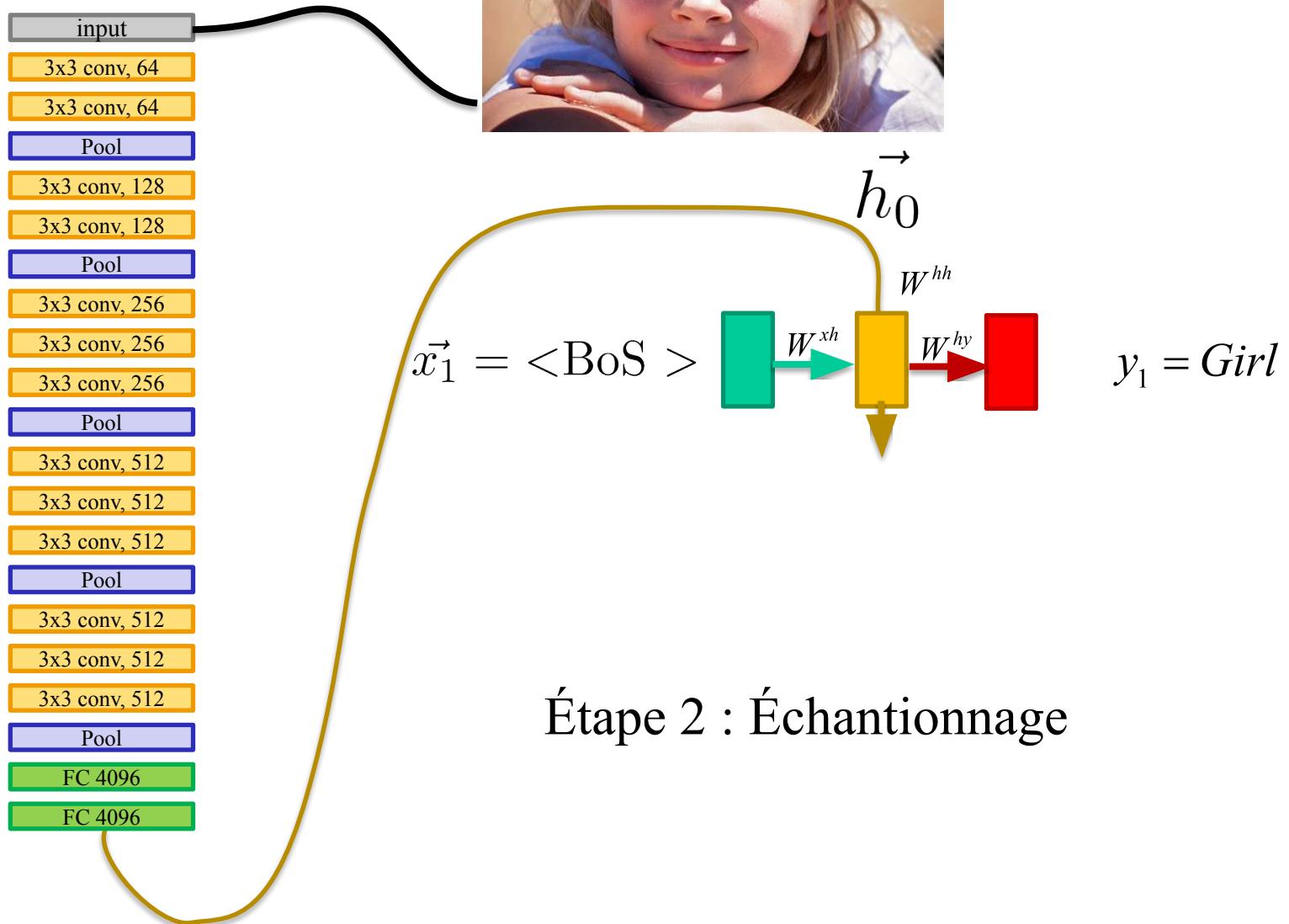


VGG16

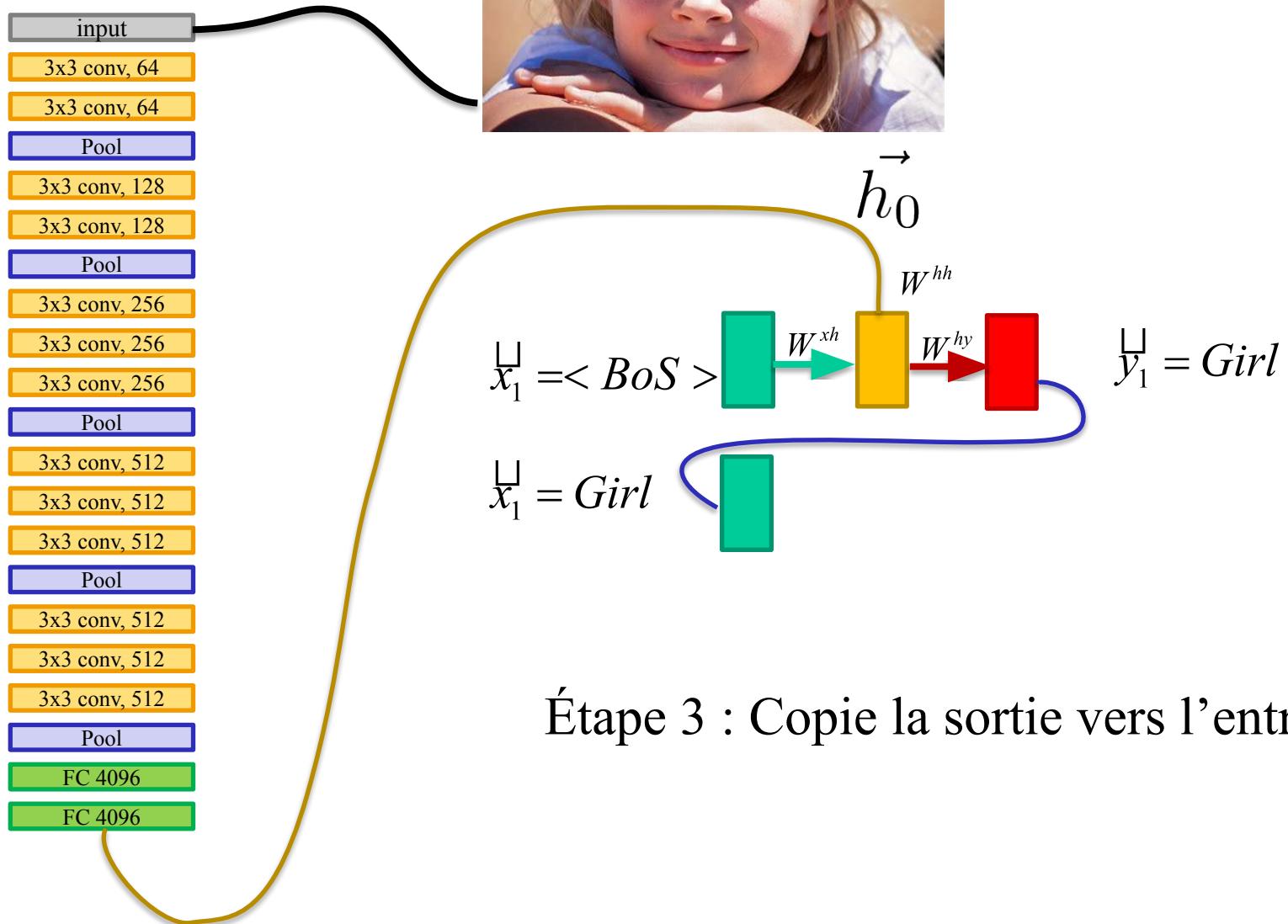
Captioning



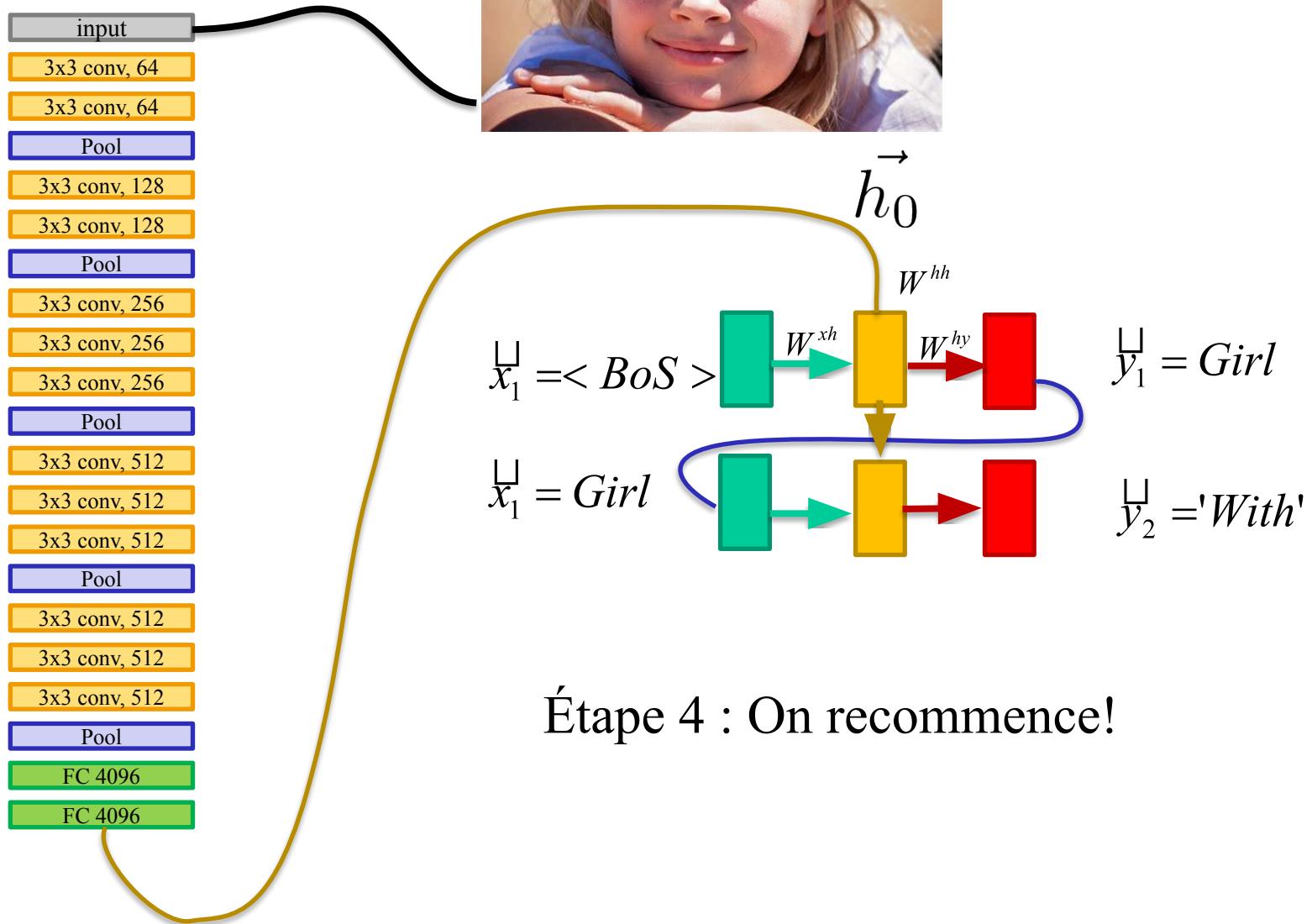
Captioning



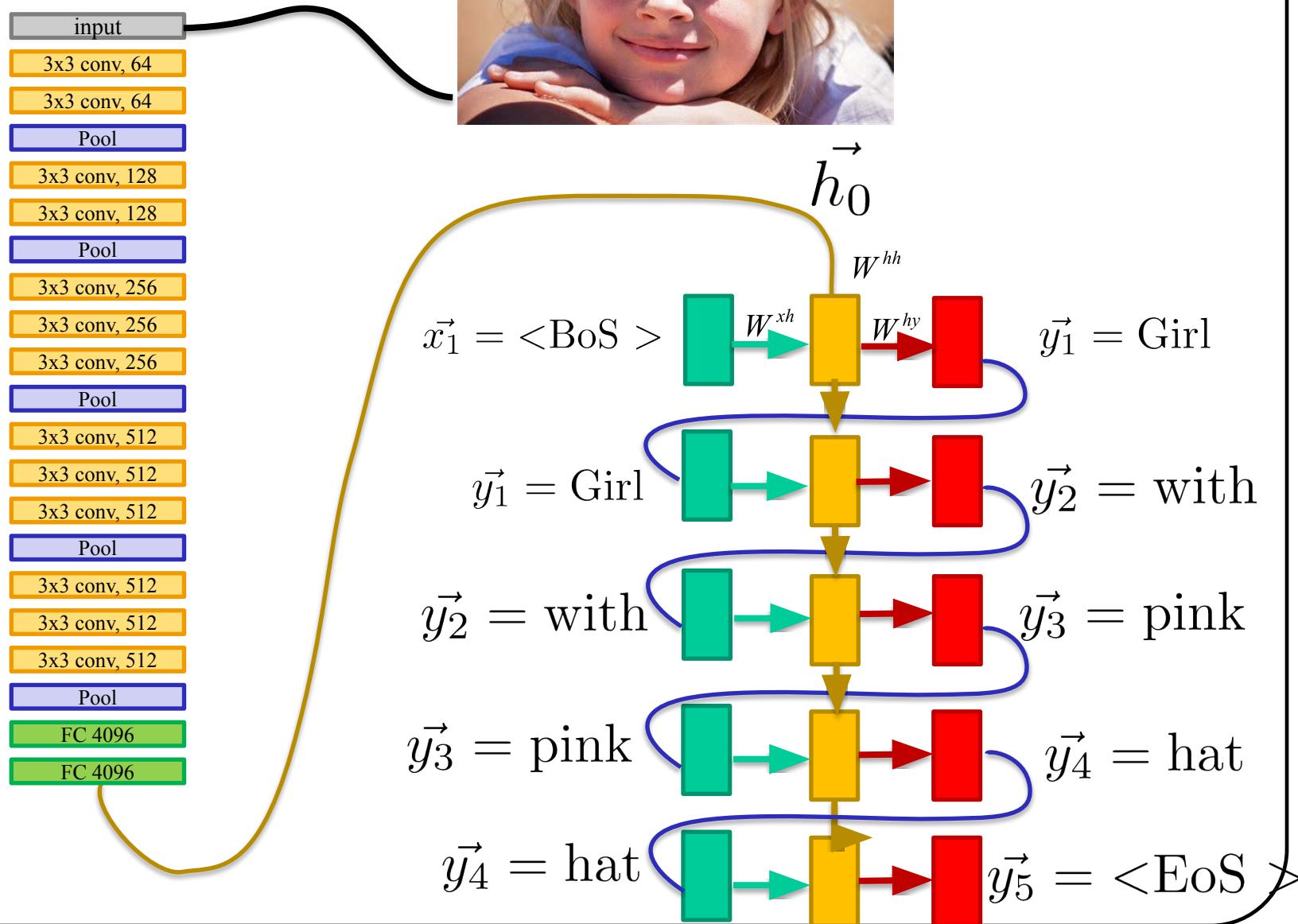
Captioning



Captioning



Captioning



Exemples de résultats

<https://github.com/karpathy/neuraltalk2>



an elephant standing in a grassy field with trees in the background



a man riding a wave on top of a surfboard



a street sign on a pole in front of a building



a group of people playing a game with nintendo wii controllers



a couple of zebra standing on top of a dirt field

Exemples d'erreurs

<https://github.com/karpathy/neuraltalk2>



a man is throwing a frisbee in a park



a man riding a skateboard down a street



a laptop computer sitting on top of a wooden desk



A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard

NeuralTalk and Walk

<https://vimeo.com/146492001>



Limites des « one-hot vectors »

Souvent les modèles de langue utilisent l'encodage « one hot »

Pour des **caractères**...

$$\begin{aligned} 'a' &= [1,0,0,\dots,0] \\ 'b' &= [0,1,0,\dots,0] \\ 'c' &= [0,0,1,\dots,0] \end{aligned} \quad \left. \right\} \in R^{256}$$

...

Limites des « one-hot vectors »

Souvent les modèles de langue utilisent l'encodage « one hot »

Pour des **mots**...

...

$$\begin{aligned} \text{'grand'} &= [...] \\ \text{'grandement'} &= [...] \\ \text{'grandeur'} &= [...] \end{aligned} \quad \left. \right\} \in R^{10,000}$$

...

Limites des « one-hot vectors »

Bien que simple, cet encodage a plusieurs **inconvénients**

1- Peu efficace en mémoire lorsque non compressé

ex.: 10,000 bits pour encoder le mot « **je** » dans une langue à 10,000 mots!

2- Pas de distance sémantique entre les codes:

Ex.

distance[one-hot('bon'), one-hot('bien')] = **distance**[one-hot('bon'), one-hot('trottoir')]

Or, on souhaiterait un **code** tel que

distance[code('bon'), code('bien')] << **distance**[code('bon'), code('trottoir')]

distance[code('Jean'), code('Chantal')] << **distance**[code('bon'), code('trottoir')]

distance[code('Inde'), code('Liban')] << **distance**[code('bon'), code('trottoir')]

Word2Vec s'appuie sur 2 idées fondamentales

Une solution est d'utiliser l'encodage **Word2Vec** de [Mikolov et al. '13]

Idée 1: Dictionnaire = matrice d'encodage

Exemple jouet: on veut représenter ces 8 mots par un code à 4 éléments

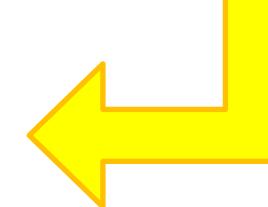
« *one-hot* »

‘the’	1	0	0	0	0	0	0	0
‘quick’	0	1	0	0	0	0	0	0
‘brown’	0	0	1	0	0	0	0	0
‘fox’	0	0	0	1	0	0	0	0
‘jumps’	0	0	0	0	1	0	0	0
‘over’	0	0	0	0	0	1	0	0
‘lazy’	0	0	0	0	0	0	1	0
‘dog’	0	0	0	0	0	0	0	1

Dictionnaire

2	3	4	5
-1	-3	-2	2
11	6	4	-3
-4	8	-4	4
24	-6	42	17
91	13	14	-5
0	36	4	56
-1	0	1	35

1 ligne = code pour 1 mot



Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage

Comment sélectionner le code d'un mot? En multipliant son vecteur One-hot par la matrice d'encodage (le dictionnaire!)

Ex: sélectionner le code de « brown »

Dictionnaire
(matrice d'encodage)

$$\left(\begin{array}{cccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \quad \left(\begin{array}{cccc} 2 & 3 & 4 & 5 \\ -1 & -3 & -2 & 2 \\ 11 & 6 & 4 & -3 \\ -4 & 8 & -4 & 4 \\ 24 & -6 & 42 & 17 \\ 91 & 13 & 14 & -5 \\ 0 & 36 & 4 & 56 \\ -1 & 0 & 1 & 35 \end{array} \right) = \left(\begin{array}{cccc} 11 & 6 & 4 & -3 \end{array} \right)$$

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage

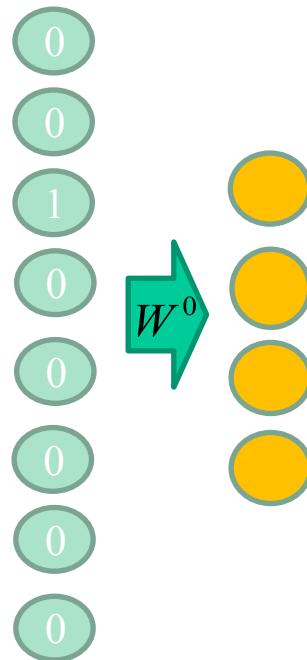
Première couche d'un réseau de neurones

=

matrice d'encodage



\vec{x} : brown



... $W^0 \in R^{4 \times 8}$

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage

Première couche d'un réseau de neurones

=

matrice d'encodage



$$code_{\vec{x}} = W^0 \vec{x}$$

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage



On pourra donc utiliser un réseau de neurones
pour calculer le contenu du dictionnaire

Word2Vec s'appuie sur 2 idées fondamentales

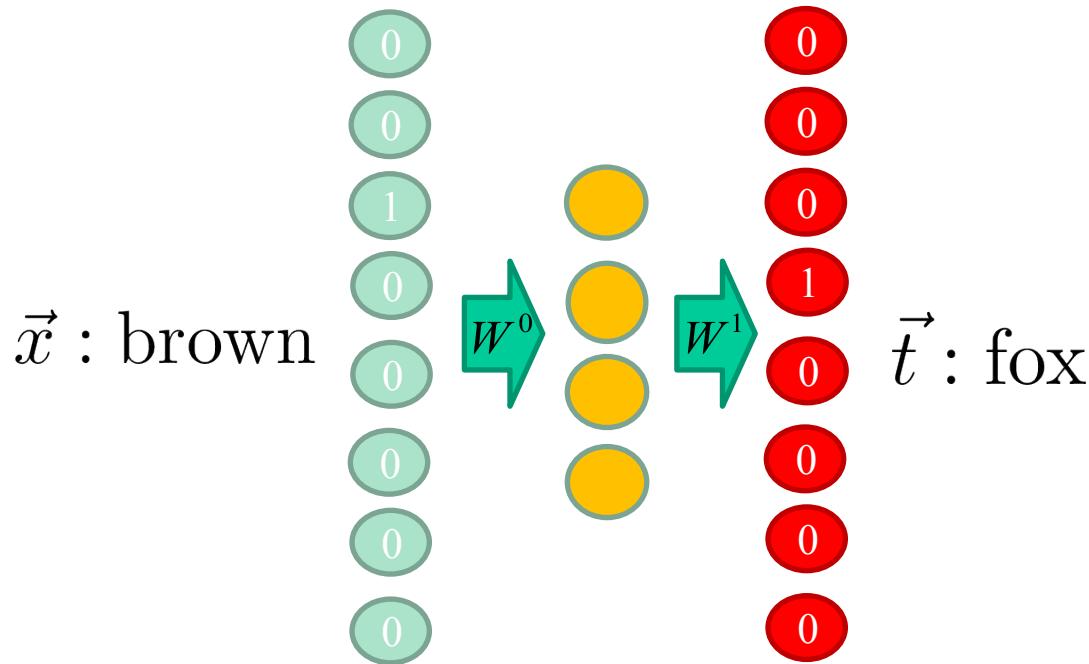
Idée 2: 2 mots proches dans un texte = 2 mots proches sémantiquement

Source Text	Training Samples
The quick brown fox jumps over the lazy dog. →	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog. →	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog. →	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog. →	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Basé sur un corpus de texte, on va créer des **millions de paires de mots**

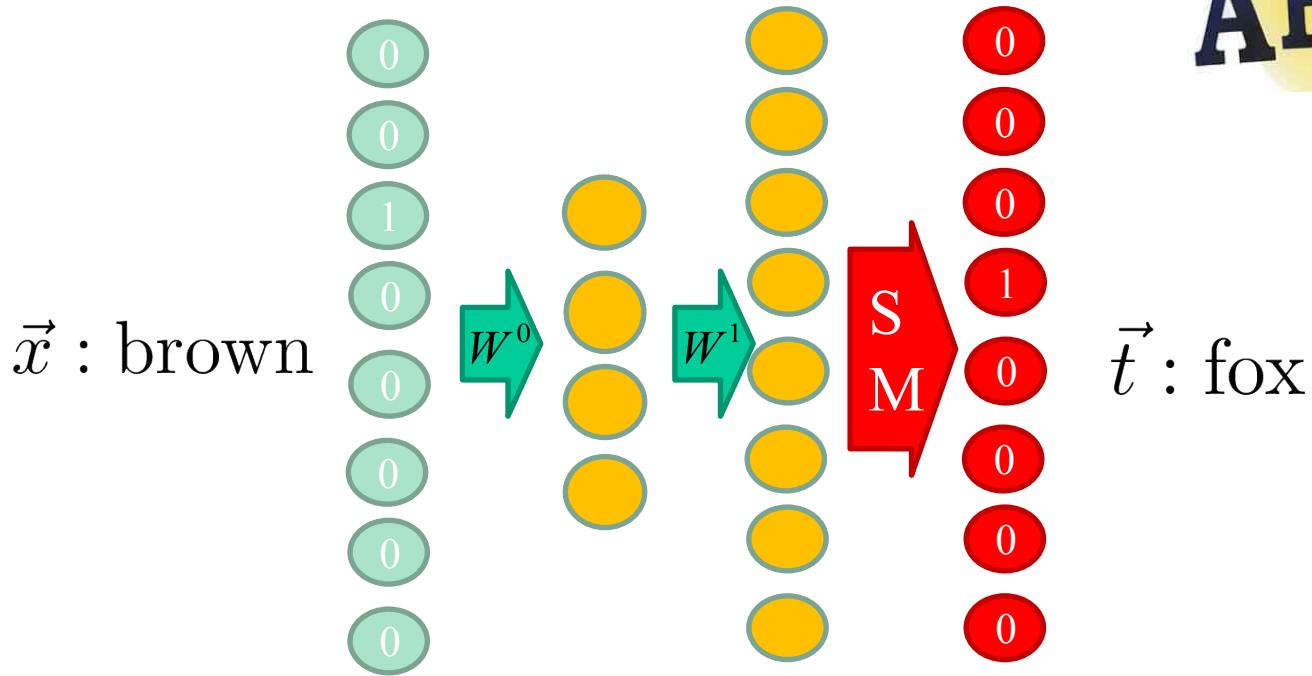
Word2Vec

[Mikolov et al. '13]



Entraîner un réseau de neurones
à reproduire le 2^e mot partant du 1^{er}

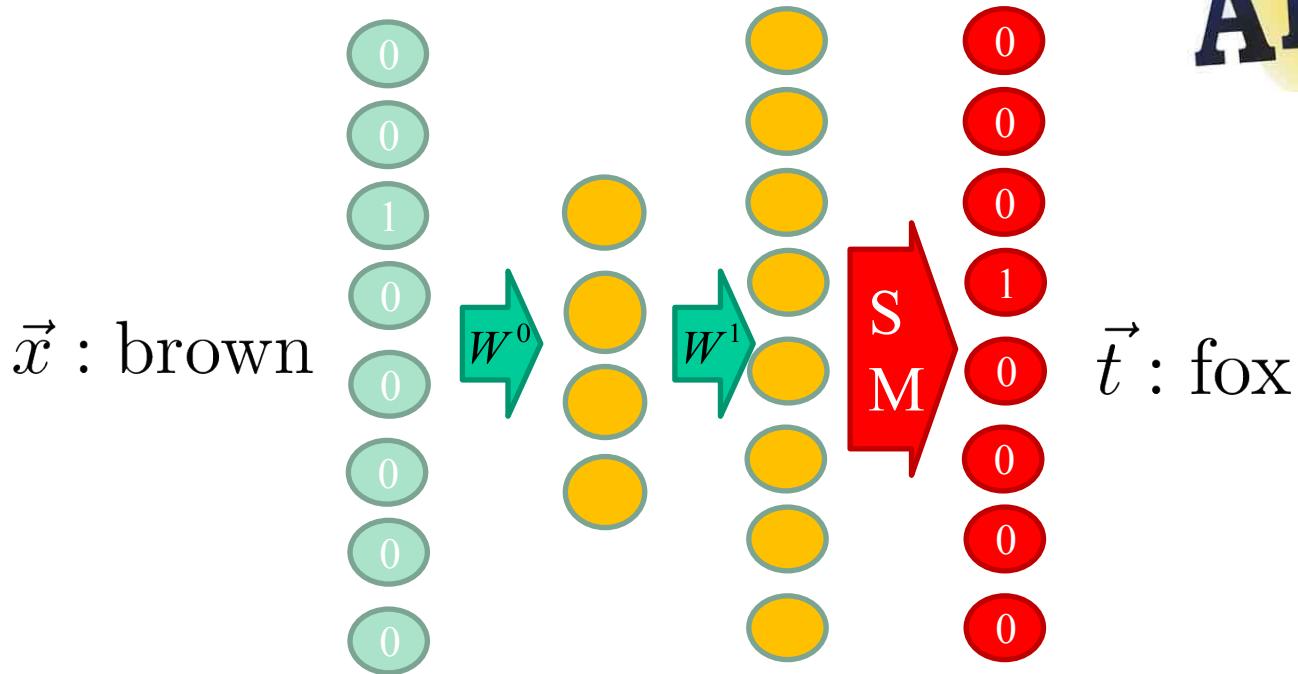
Word2Vec [Mikolov et al. '13]



Puisque la sortie est de type « *one-hot* »
on utilise un softmax

Word2Vec

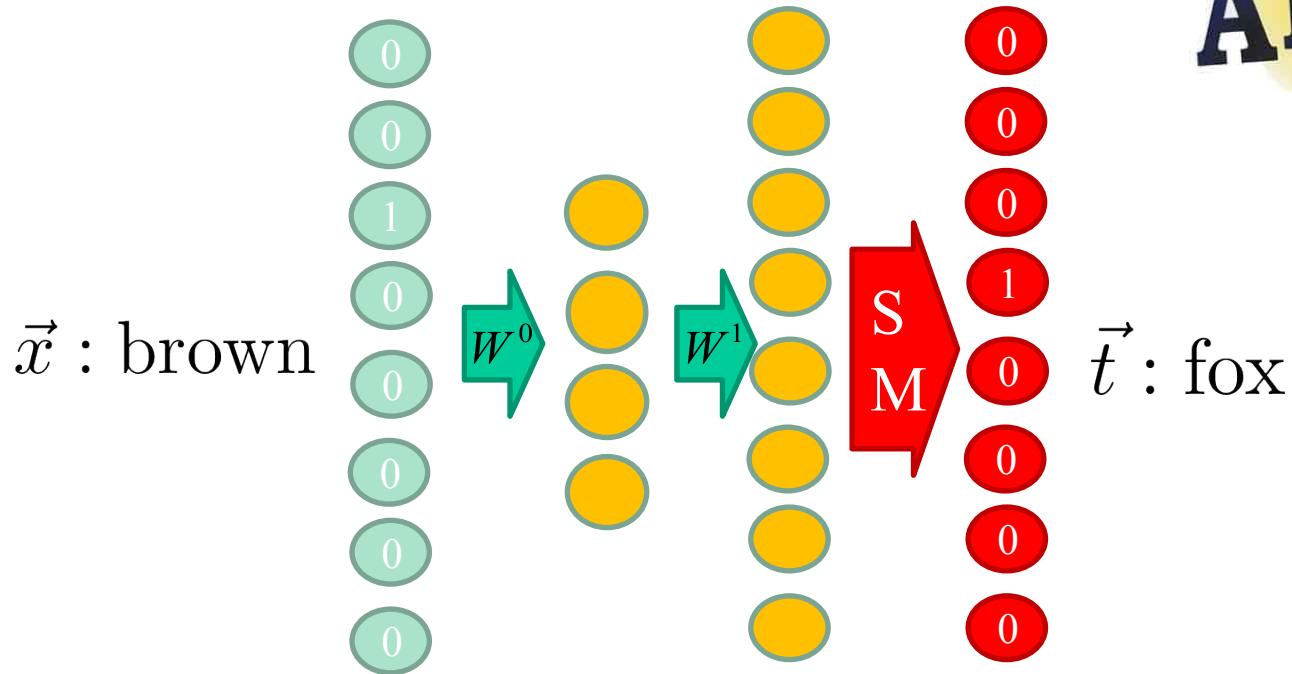
[Mikolov et al. '13]



$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0 \mathbf{x}))$$

Word2Vec

[Mikolov et al. '13]



Lorsqu'entraîné, utiliser W^0
comme dictionnaire

Word2Vec [Mikolov et al. '13]

“If two different words have very similar “contexts” (that is, what words are likely to appear around them), then our model needs to output very similar results for these two words. And one way for the network to output similar context predictions for these two words is if the word vectors are similar. So, if two words have similar contexts, then our network is motivated to learn similar word vectors for these two words! Ta da!

And what does it mean for two words to have similar contexts? I think you could expect that synonyms like “intelligent” and “smart” would have very similar contexts. Or that words that are related, like “engine” and “transmission”, would probably have similar contexts as well.”

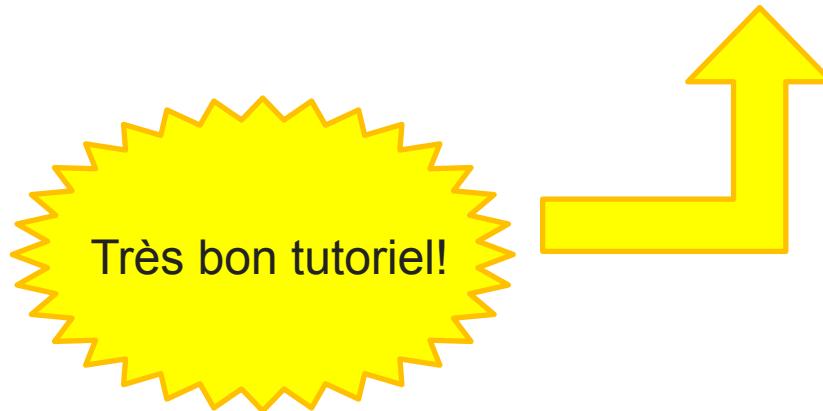
Word2Vec [Mikolov et al. '13]

Cet algorithme vient avec **d'autres détails**

- Réduire l'occurrence des mots fréquents et sémantiquement faibles (*the, of, for, this, or, and, ...*)
- Combiner des mots qui forment une entité (ex: *nations unies*)
- Divers trucs pour simplifier/accélérer l'entraînement

Limites du « one-hot vector »

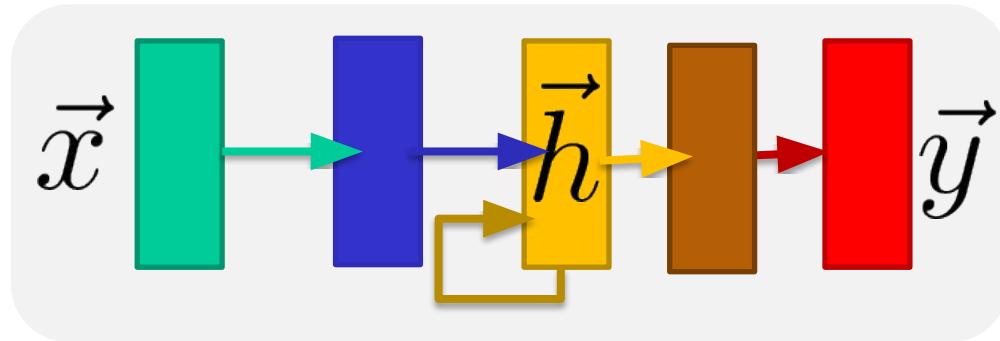
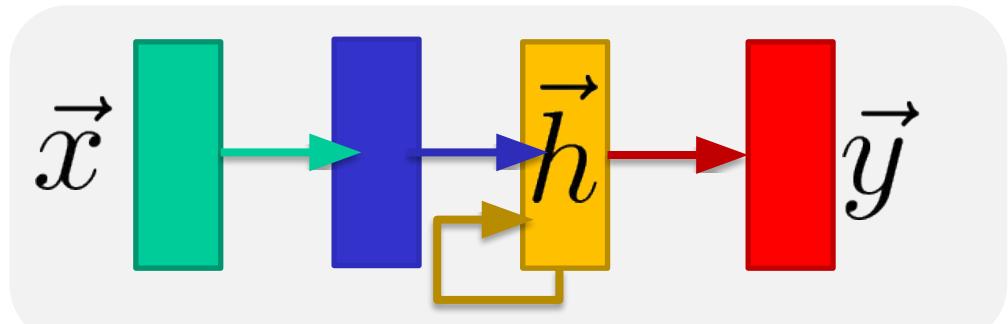
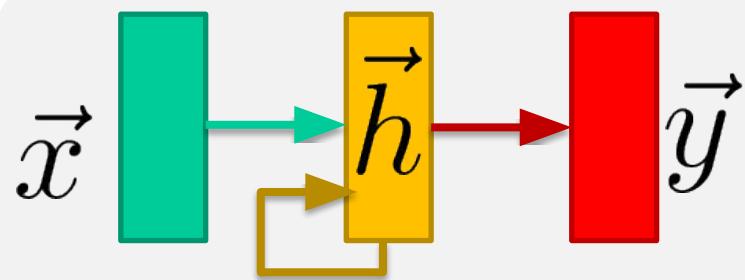
<http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>



T.Mikolov et al. (2013). "Efficient Estimation of Word Representations in Vector Space", in ICLR 2013

Encodage (*embedding*)

L'encodage peut être appris !



La **couche d'encodage** permet de faire la sélection implicitement et d'ajouter plus de transformations

Prédiction sur des lettres vs. mots

$$\begin{aligned}'a' &= [1,0,0,\dots,0] \\'b' &= [0,1,0,\dots,0] \\'c' &= [0,0,1,\dots,0]\end{aligned}\quad \left.\right\} \in R^{256}$$

Prédiction sur des lettres

...

...

$$\begin{aligned}'grand' &= [...,1,0,0,\dots,0] \\'grandement' &= [...,0,1,0,\dots,0] \\'grandeur' &= [...,0,0,1,\dots,0]\end{aligned}\quad \left.\right\} \in R^{10,000}$$

Prédiction sur des mots

...

Prédiction sur des lettres vs. mots

$$\begin{aligned} \text{'e'} &= [0, 0, \dots, 1, \dots, 0] \\ &\quad \cdots \\ \text{'grand'} &= [0, 0, \dots, 1, \dots, 0] \\ &\quad \cdots \\ \text{'ment'} &= [0, 0, \dots, 1, \dots, 0] \\ &\quad \cdots \end{aligned} \quad \left. \right\} \in \mathbb{R}^m$$

Prédiction sur les deux !

‘grand’
‘grand’+‘e’
‘grand’+‘e’+‘ment’

Tokenization (jeton-isation ?)

Idée: à partir d'un dictionnaire qui ne contient que des caractères, combiner les séquences fréquentes en jetons (*tokens*)

Les séquences fréquentes (comme les mots ou sous-mots fréquents) se voient attribuer un jeton. Les séquences peu fréquentes peuvent être bâties à partir de jetons.

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909.

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?<!S)' + bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(''.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

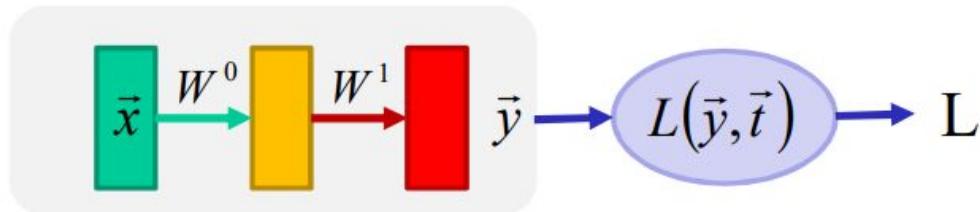
vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    vocab = merge_vocab(best, vocab)
    print(best)
```

r ·	→	r ·
l o	→	lo
l o w	→	low
e r ·	→	er ·

Comment entraîner un RNN?

Histoire de gradients

RN de classification avec entropie croisée

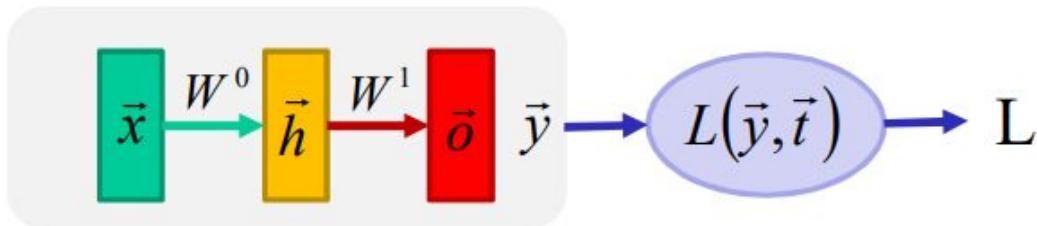


$$\vec{y}(\vec{x}) = S_M \left(W^1 \tanh \left(W^0 \vec{x} \right) \right)$$

$$L = L_{EC} \left(\vec{y}, \vec{t} \right)$$

Histoire de gradients

Simple RN de classification avec entropie croisée



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

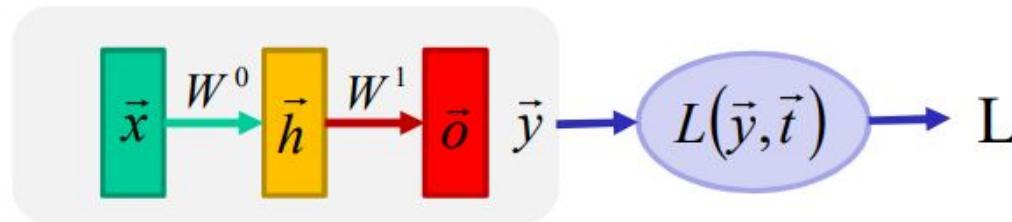
$$L = L_{CE}(\vec{y}, \vec{t})$$



Propagation
avant

Histoire de gradients

Simple RN de classification avec entropie croisée



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

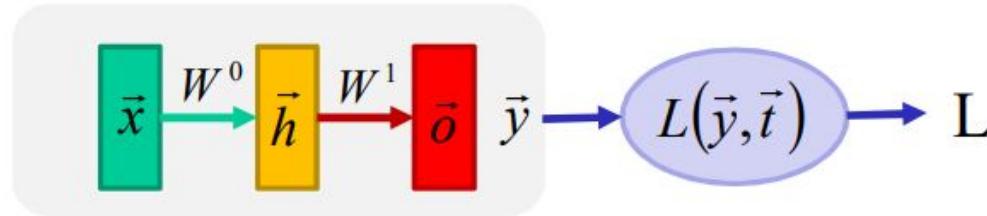
$$L = L_{CE}(\vec{y}, \vec{t})$$

Pour entraîner le réseau
il faut calculer

$$\nabla_{W^0} L \text{ et } \nabla_{W^1} L$$

Histoire de gradients

Simple RN de classification avec entropie croisée



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

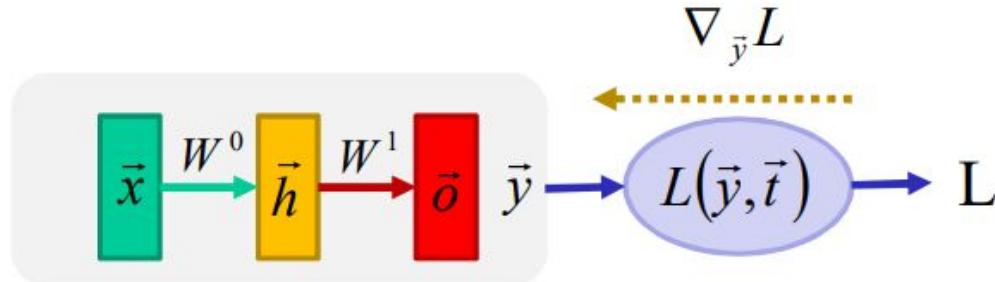
$$L = L_{CE}(\vec{y}, \vec{t})$$

Dérivée en chaîne

$$\nabla_{W^1} L = \nabla_{\vec{y}} L \nabla_{\vec{o}} \vec{y} \nabla_{W^1} \vec{o}$$

$$\nabla_{W^0} L = \nabla_{\vec{y}} L \nabla_{\vec{o}} \vec{y} \nabla_{\vec{h}} \vec{o} \nabla_{W^0} \vec{h}$$

Histoire de gradients

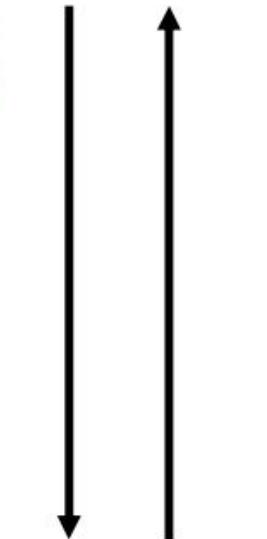


$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

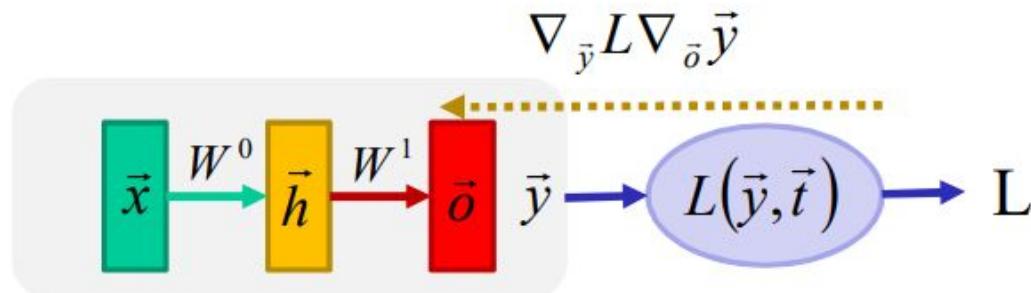
$$L = L_{CE}(\vec{y}, \vec{t})$$



$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

Histoire de gradients



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

$$L = L_{CE}(\vec{y}, \vec{t})$$

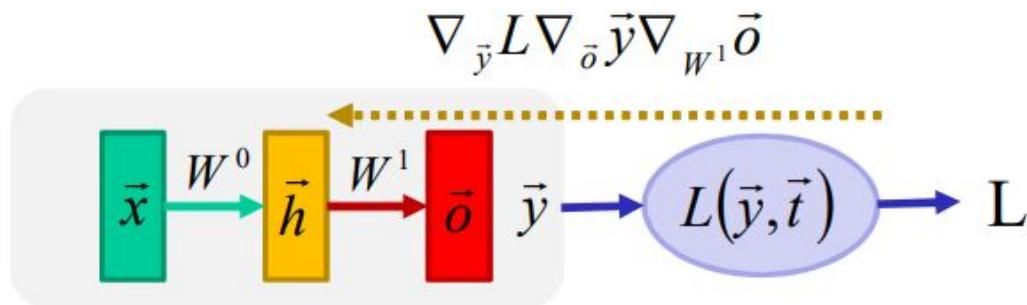


$$\nabla_{\vec{o}} \vec{y} = \mathbf{I} \vec{y}^T - \vec{t} \vec{y}$$

$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

Histoire de gradients



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

$$L = L_{CE}(\vec{y}, \vec{t})$$

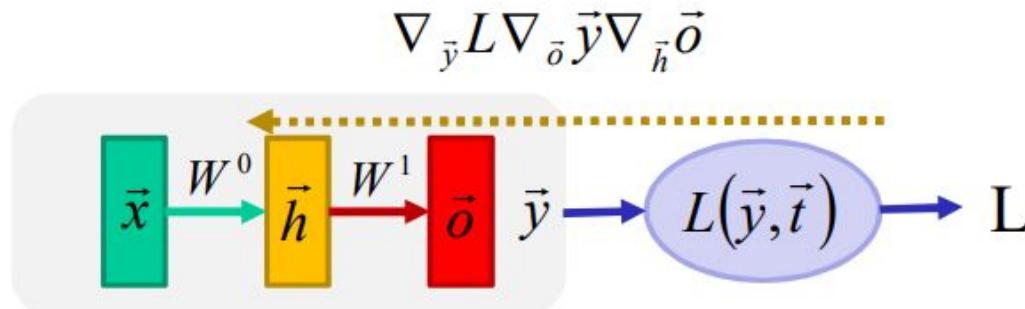
$$\nabla_{W^1} \vec{o} = \vec{h}$$

$$\nabla_{\vec{o}} \vec{y} = \mathbf{I} \vec{y}^T - \vec{y}^T \vec{y}$$

$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

Histoire de gradients



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

$$L = L_{CE}(\vec{y}, \vec{t})$$



$$\nabla_{\vec{h}} \vec{o} = W^1$$

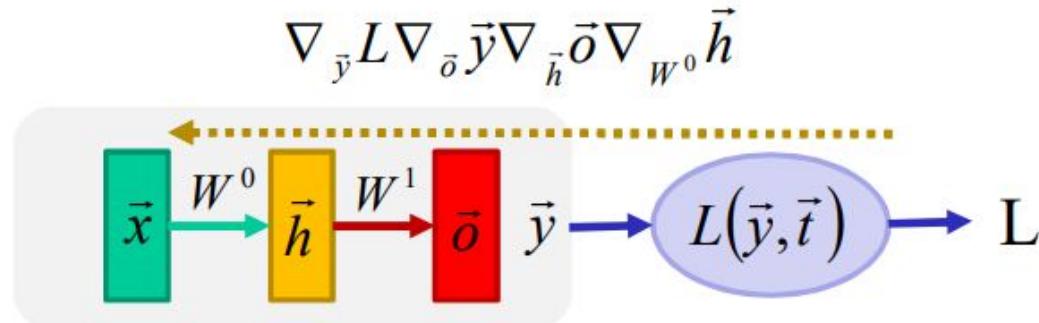
$$\nabla_{W^1} \vec{o} = \vec{h}$$

$$\nabla_{\vec{o}} \vec{y} = \mathbf{I} \vec{y}^T - \vec{y}^T \vec{y}$$

$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

Histoire de gradients



$$\vec{h} = \tanh(W^0 \vec{x})$$

$$\vec{o} = W^1 \vec{h}$$

$$\vec{y} = S_M(\vec{o})$$

$$L = L_{CE}(\vec{y}, \vec{t})$$



Rétro-
propagation

$$\nabla_{W^0} \vec{h} = 1 - \tanh^2(W^0 \vec{x}) \vec{x}$$

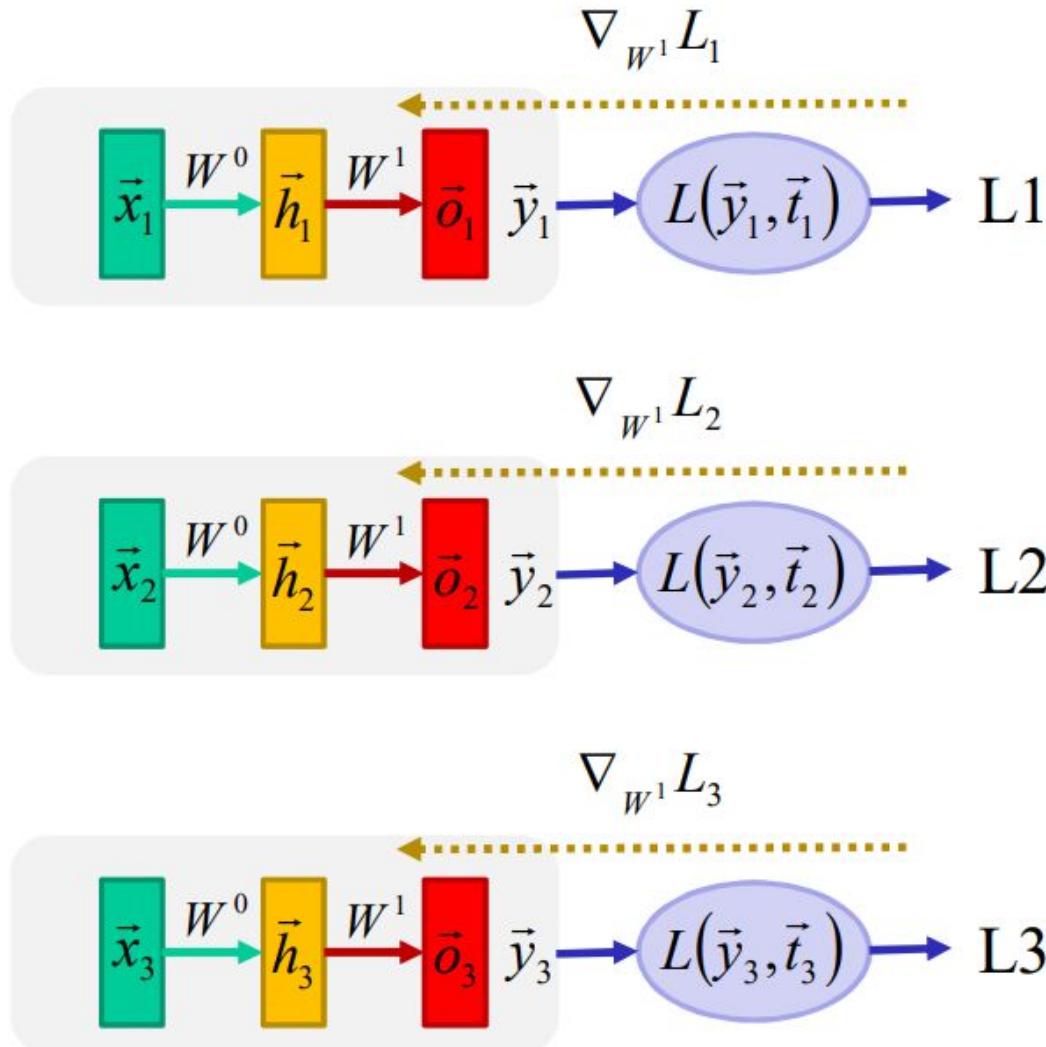
$$\nabla_{\vec{h}} \vec{o} = W^1$$

$$\nabla_{W^1} \vec{o} = \vec{h}$$

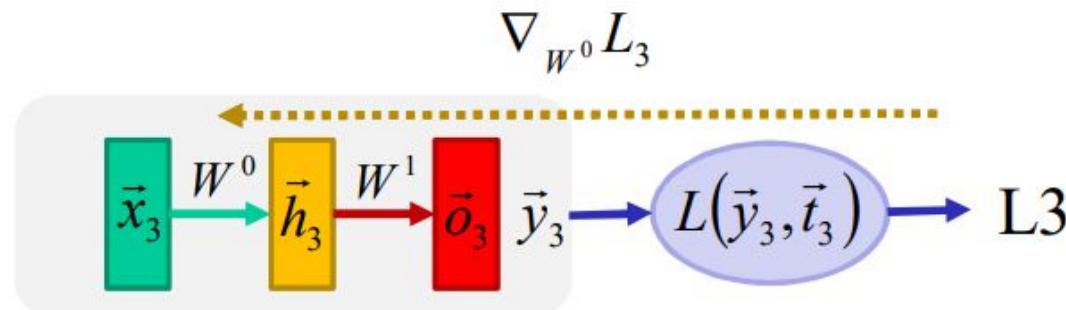
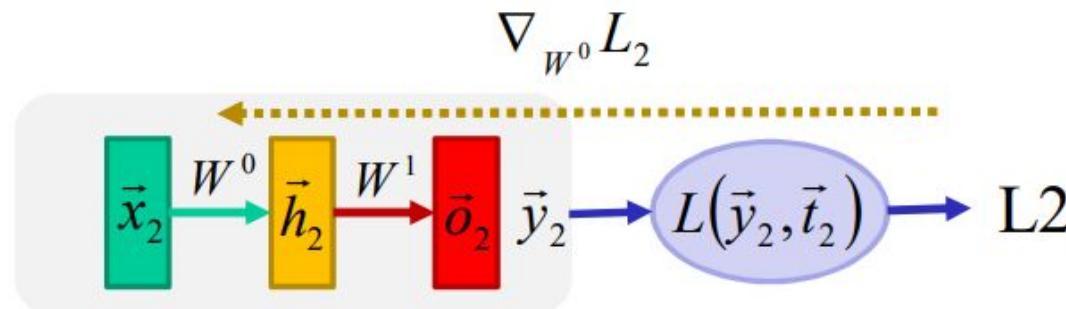
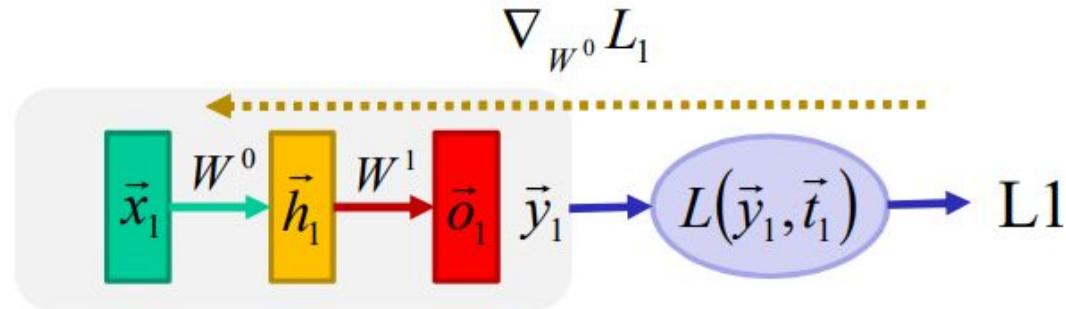
$$\nabla_{\vec{o}} \vec{y} = \mathbf{I} \vec{y}^T - \vec{y}^T \vec{y}$$

$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

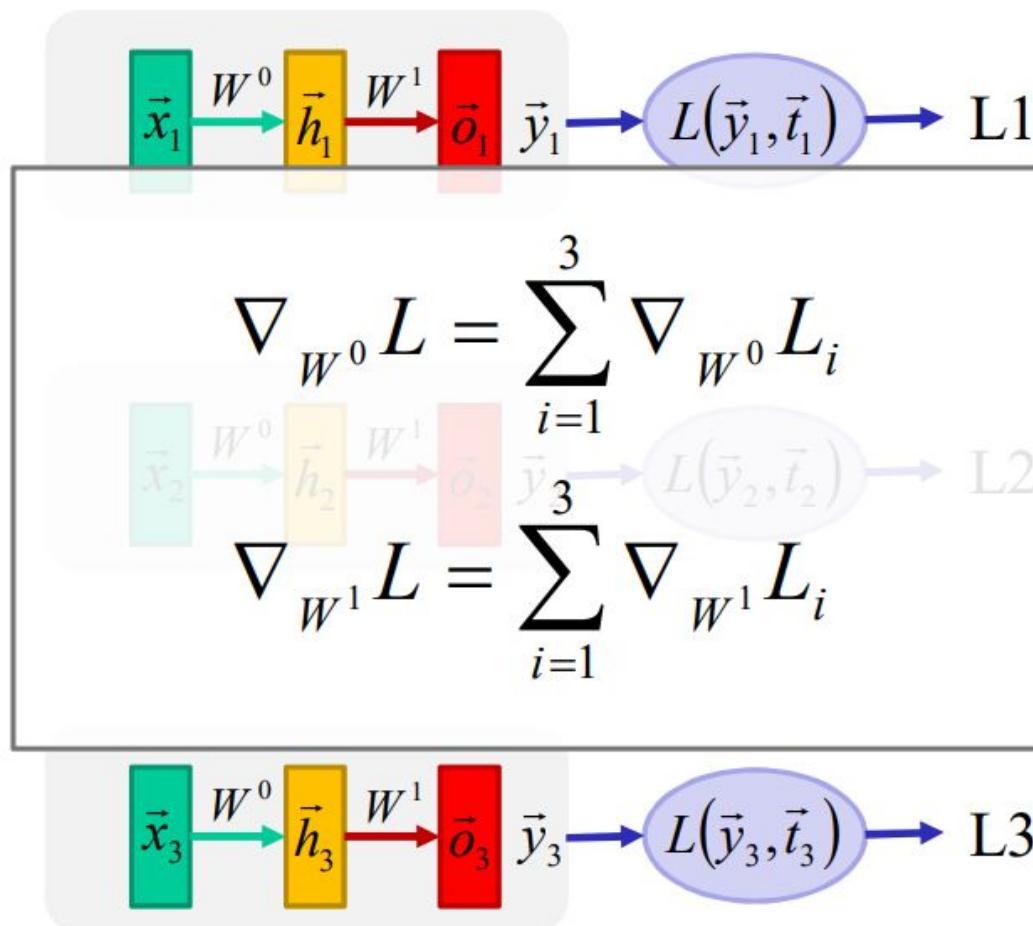
Ex.: 3 données, 3 rétro-propagations



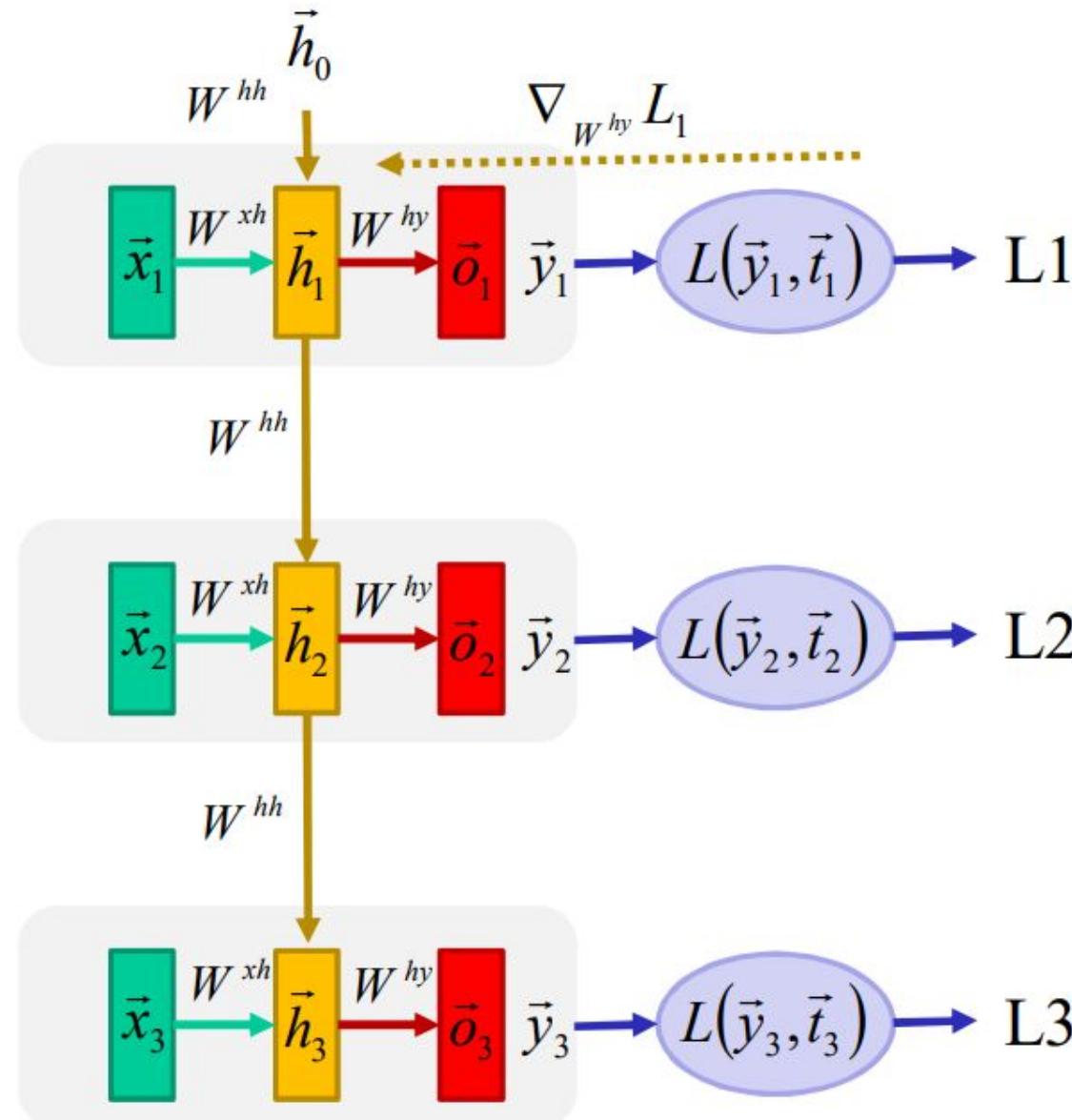
Ex.: 3 données, 3 rétro-propagations



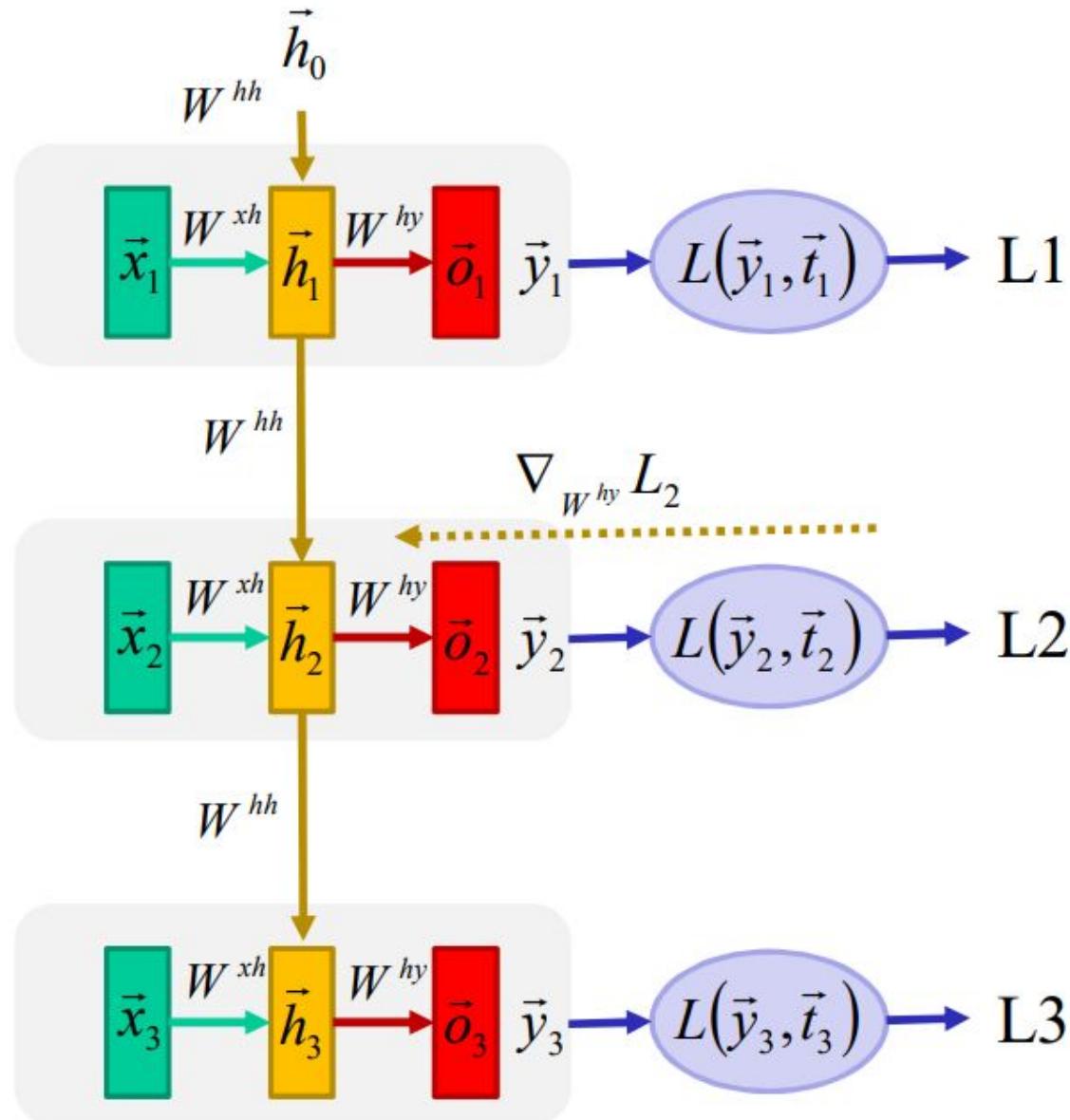
3 rétro-propagations



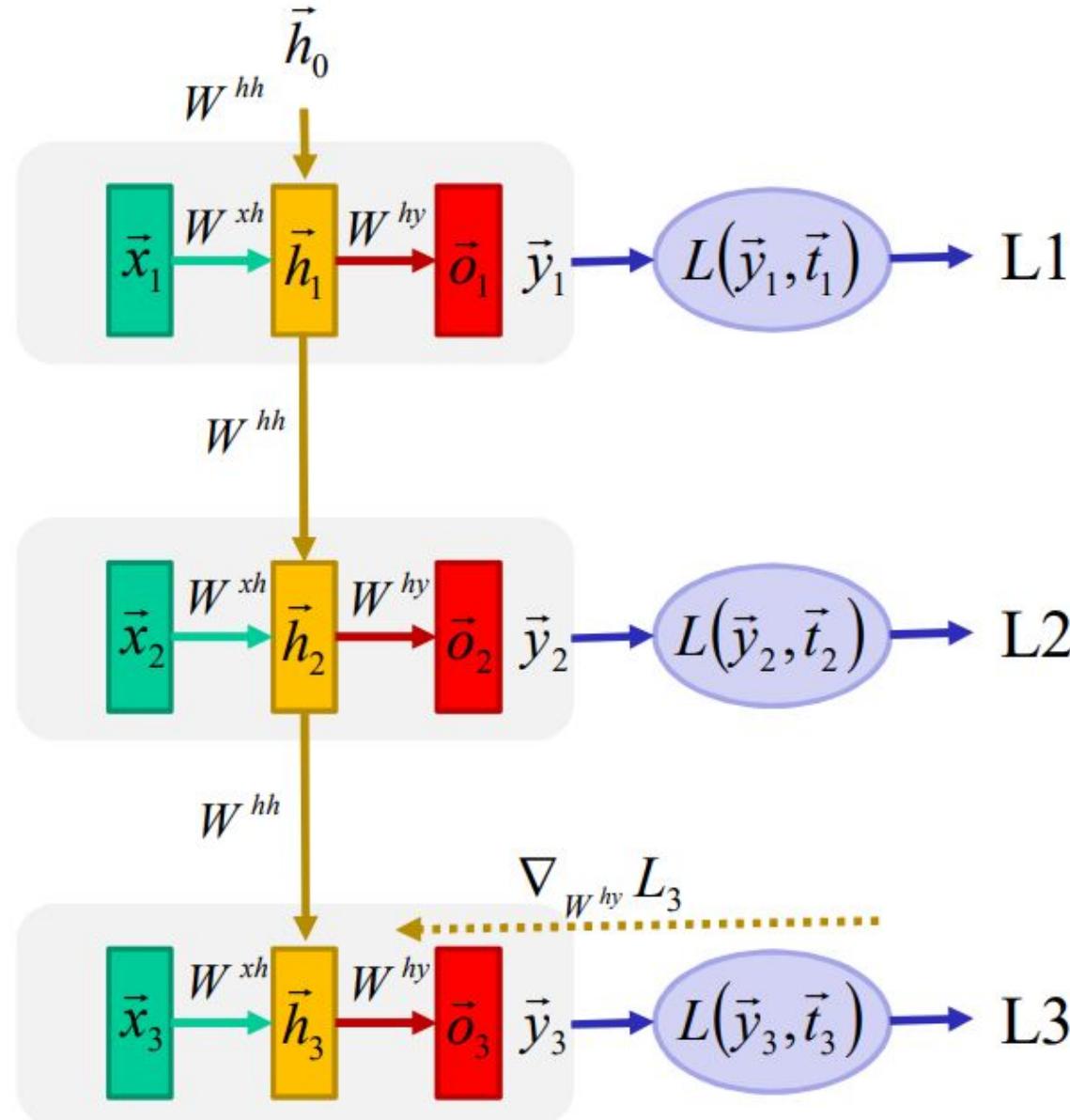
Réseau récurrent: gradient pour W^{hy}



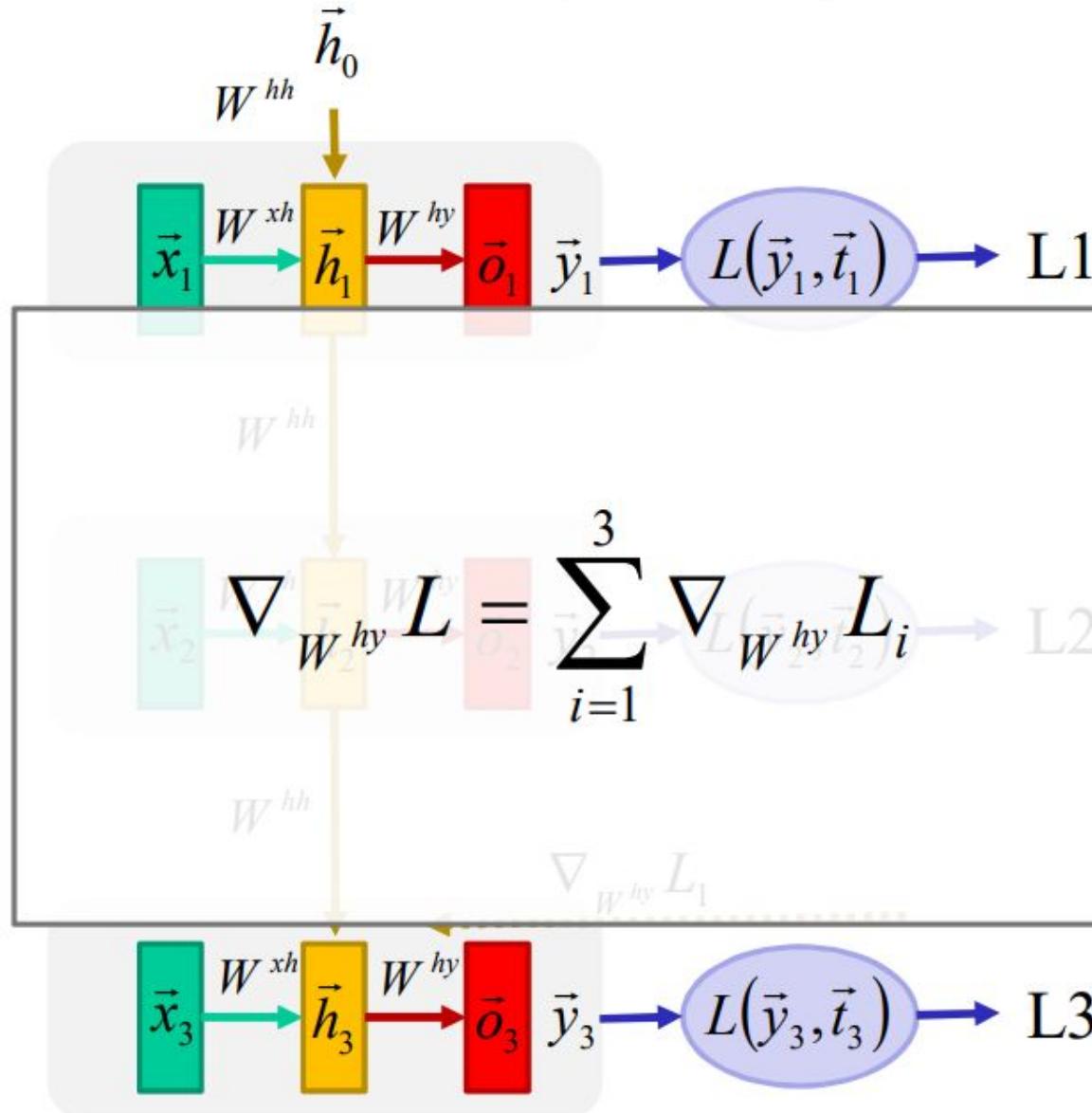
Réseau récurrent: gradient pour W^{hy}



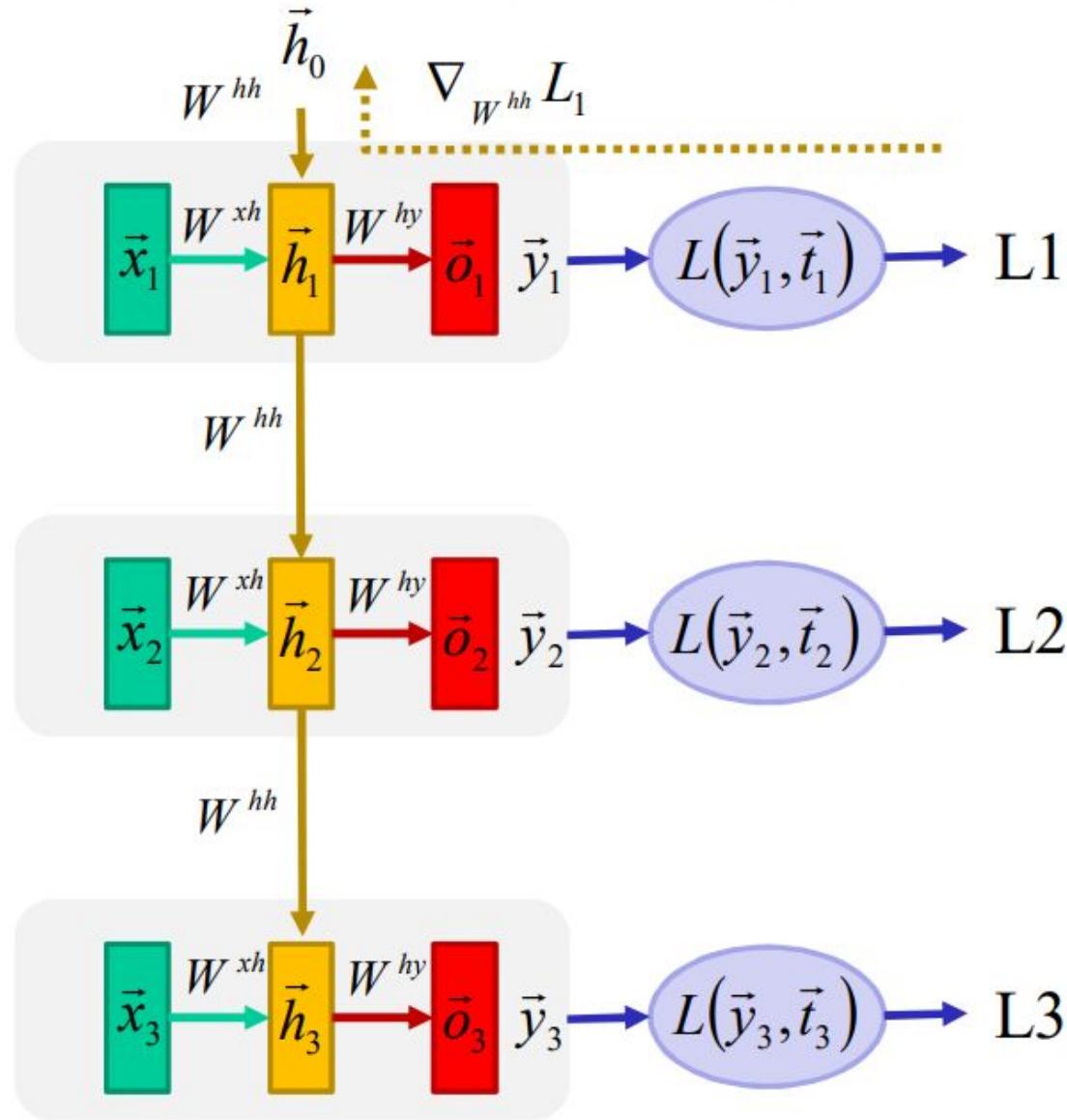
Réseau récurrent: gradient pour W^{hy}



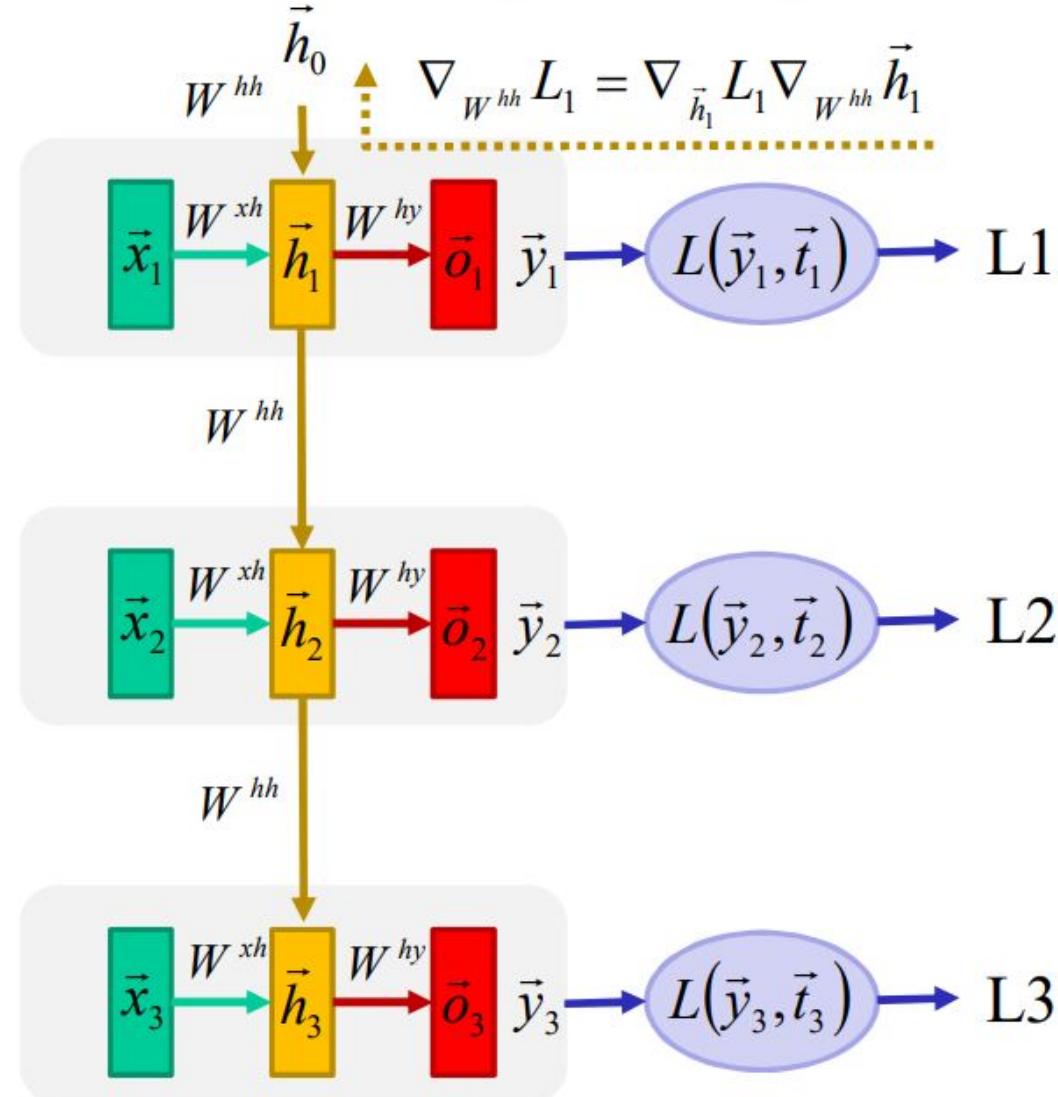
Réseau récurrent: gradient pour W^{hy}



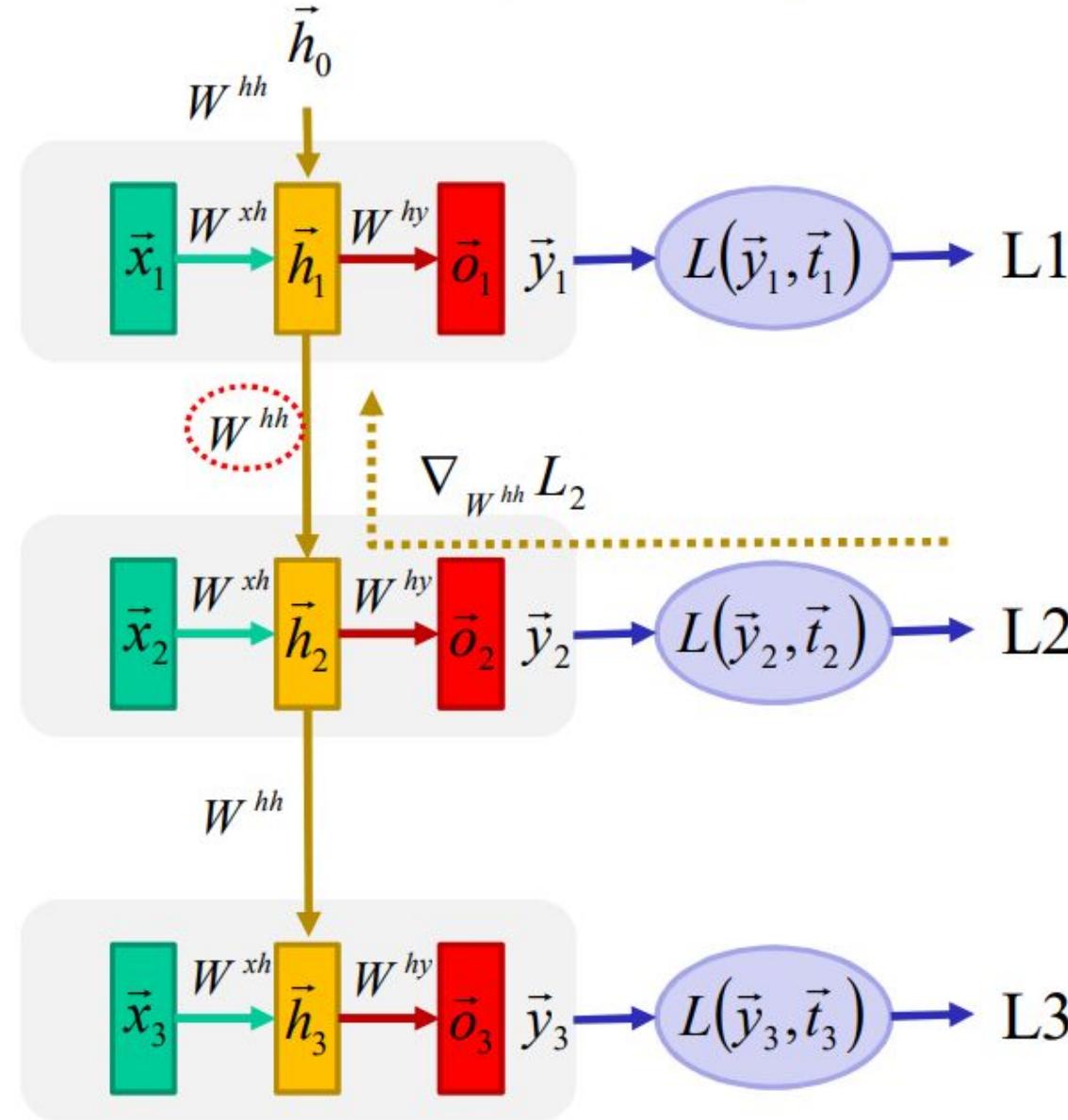
Réseau récurrent: gradient pour W^{hh}



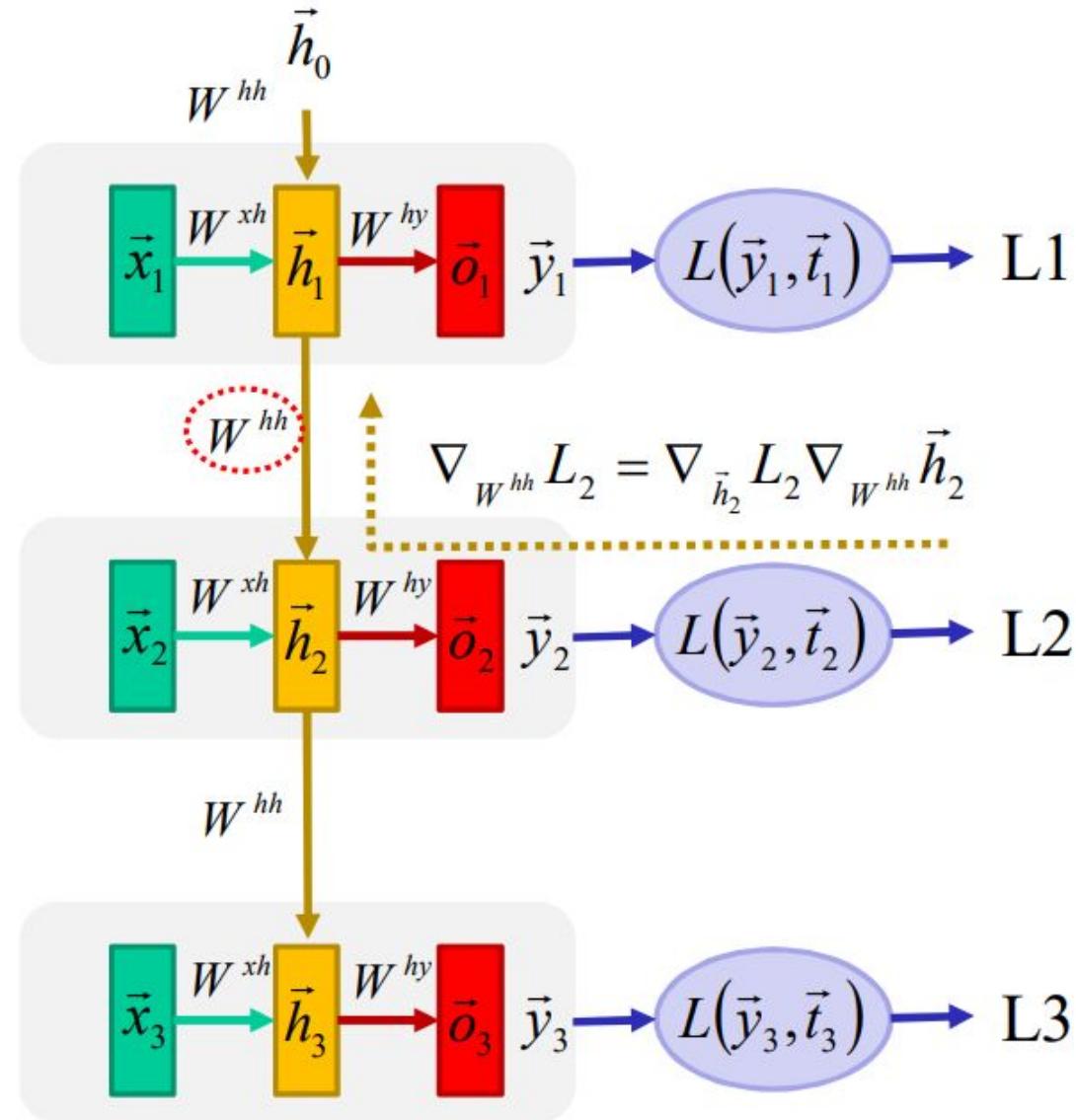
Réseau récurrent: gradient pour W^{hh}



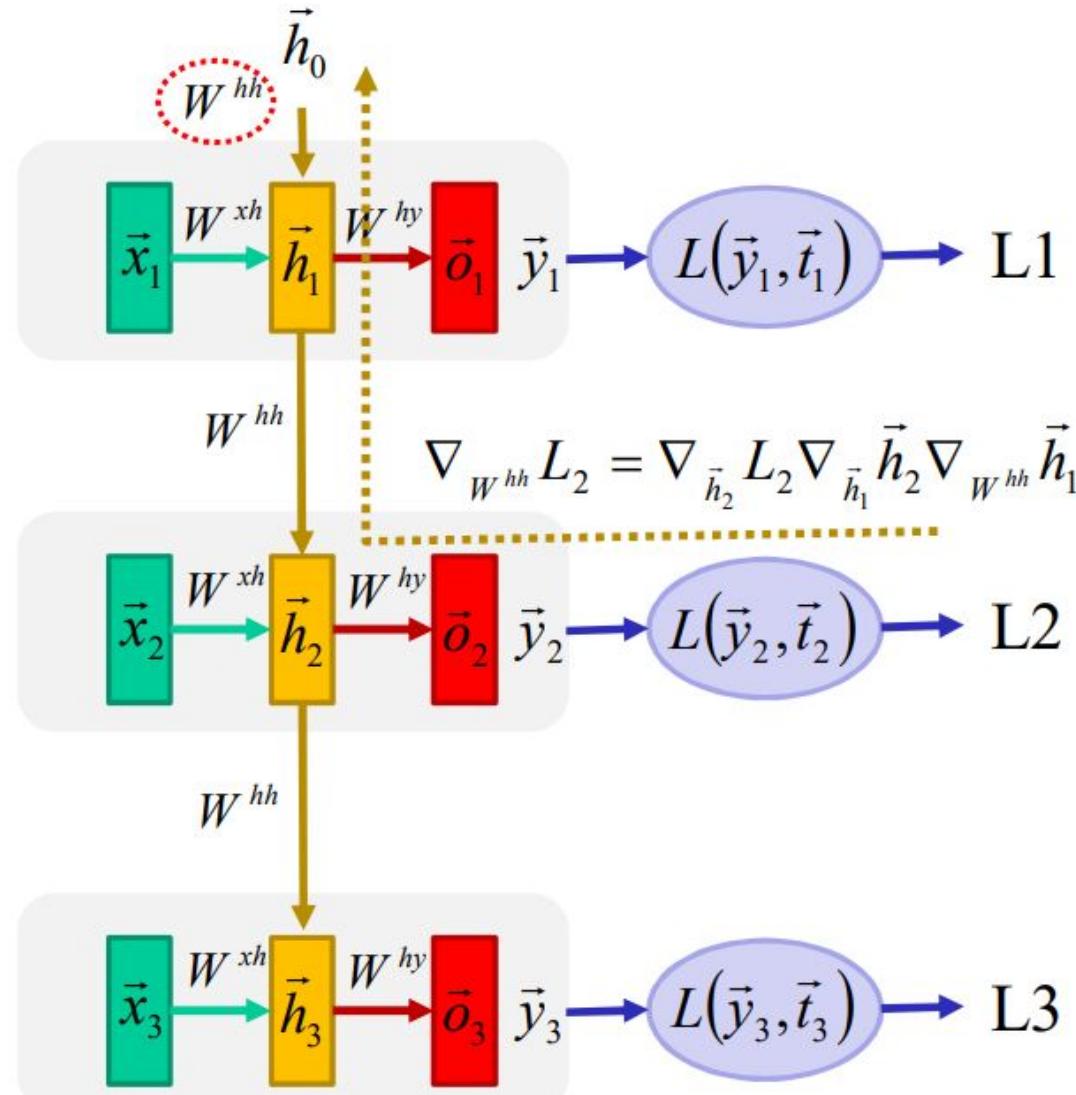
Réseau récurrent: gradient pour W^{hh}



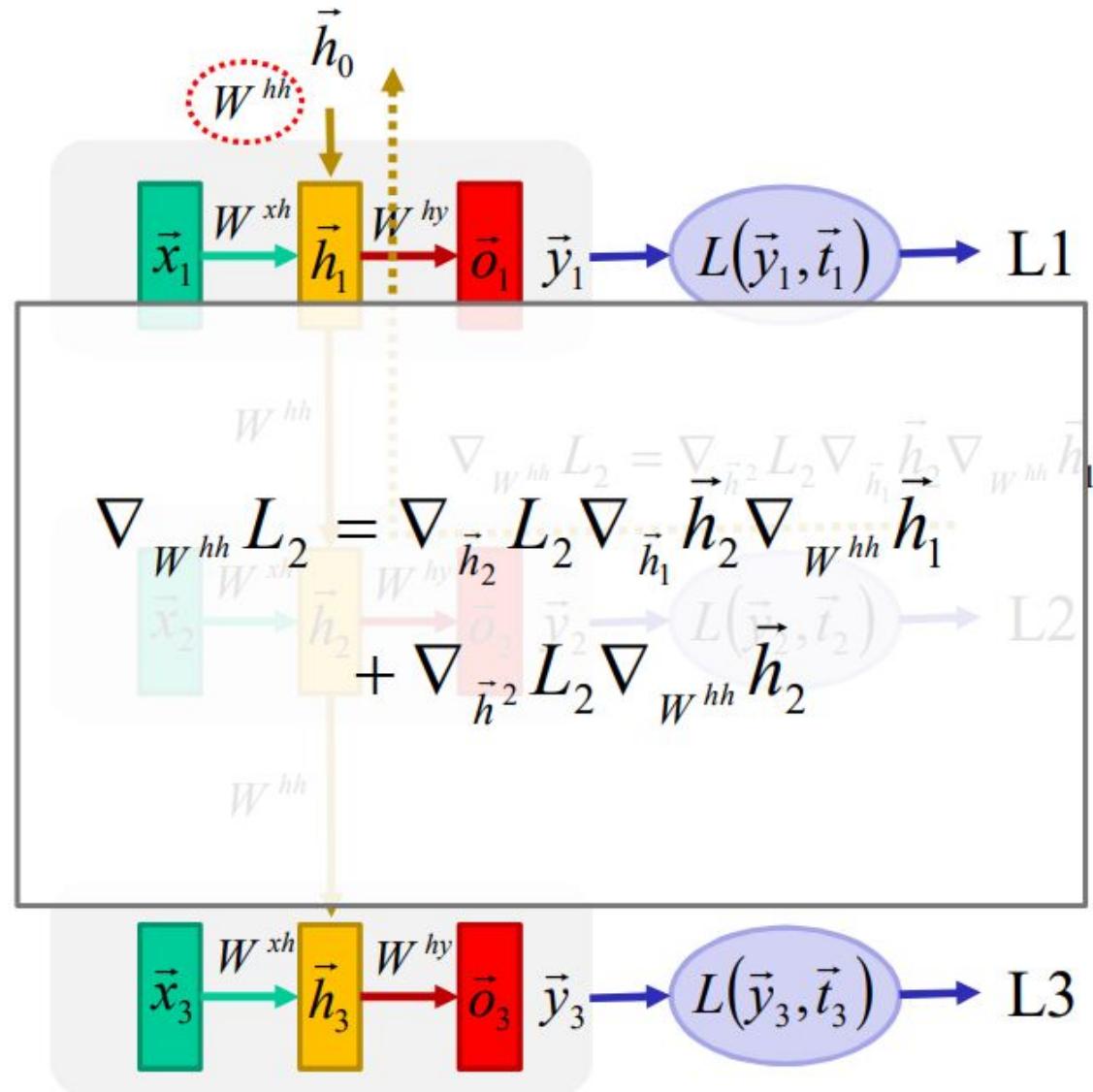
Réseau récurrent: gradient pour W^{hh}



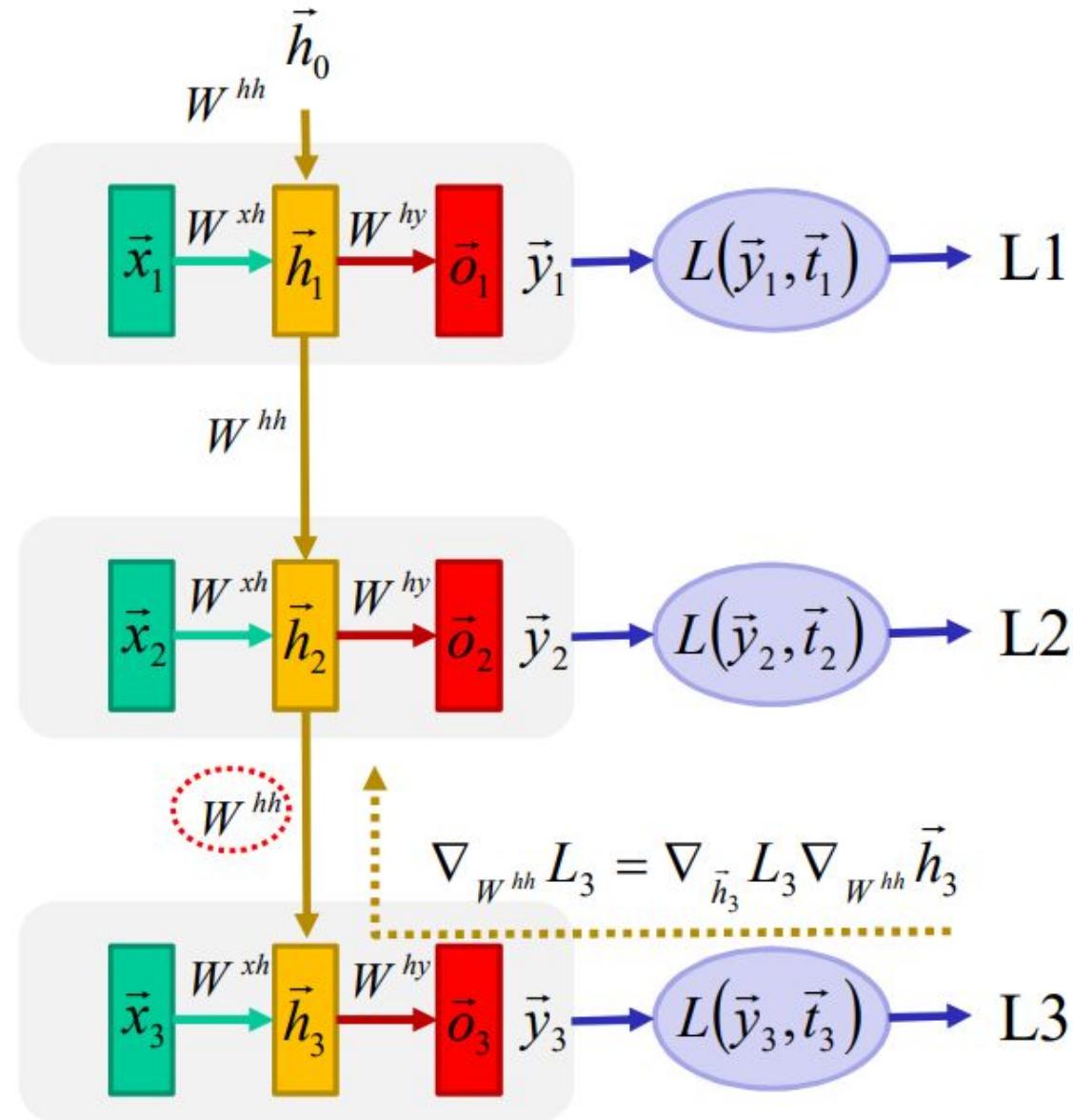
Réseau récurrent: gradient pour W^{hh}



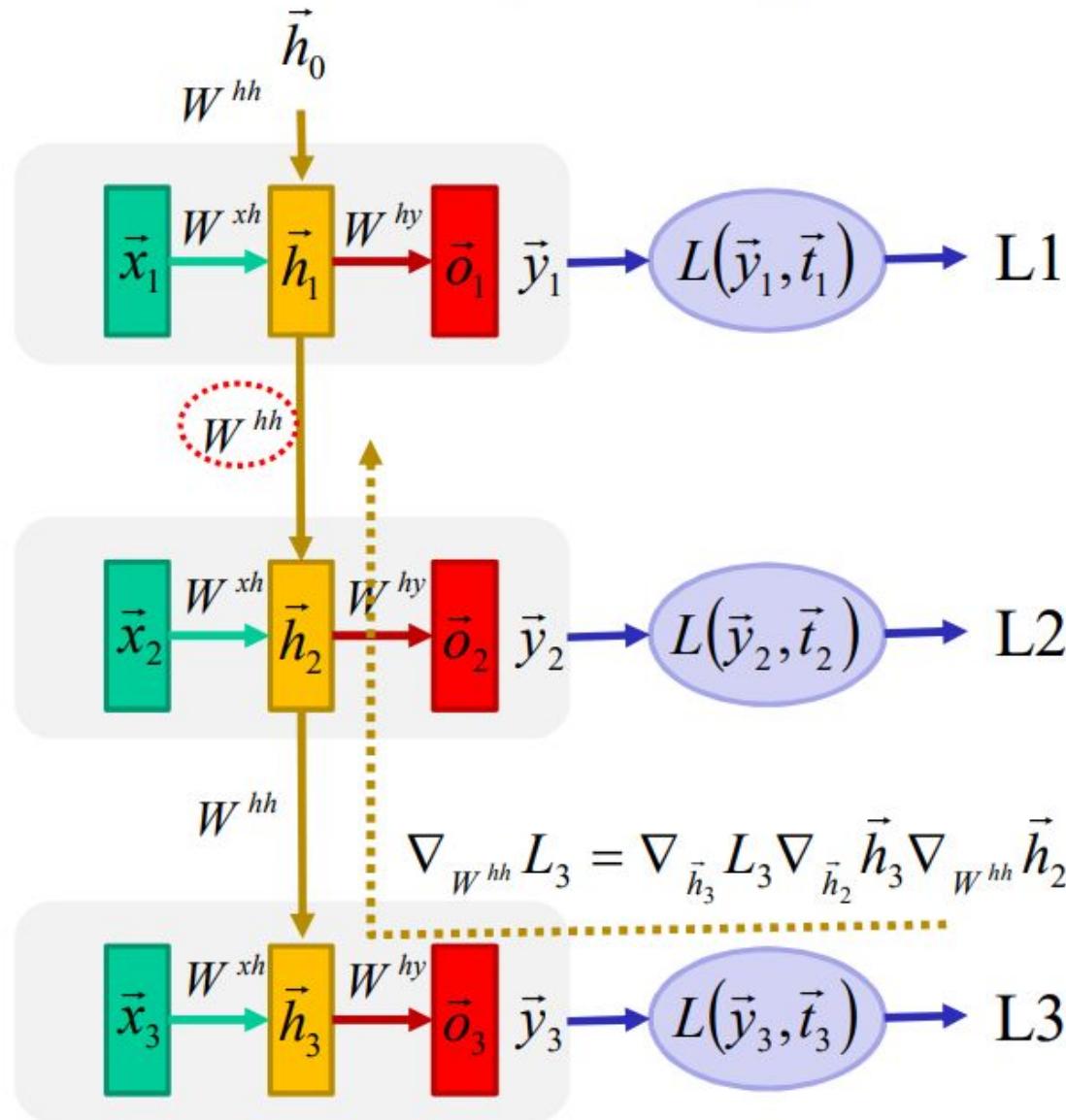
Réseau récurrent: gradient pour W^{hh}



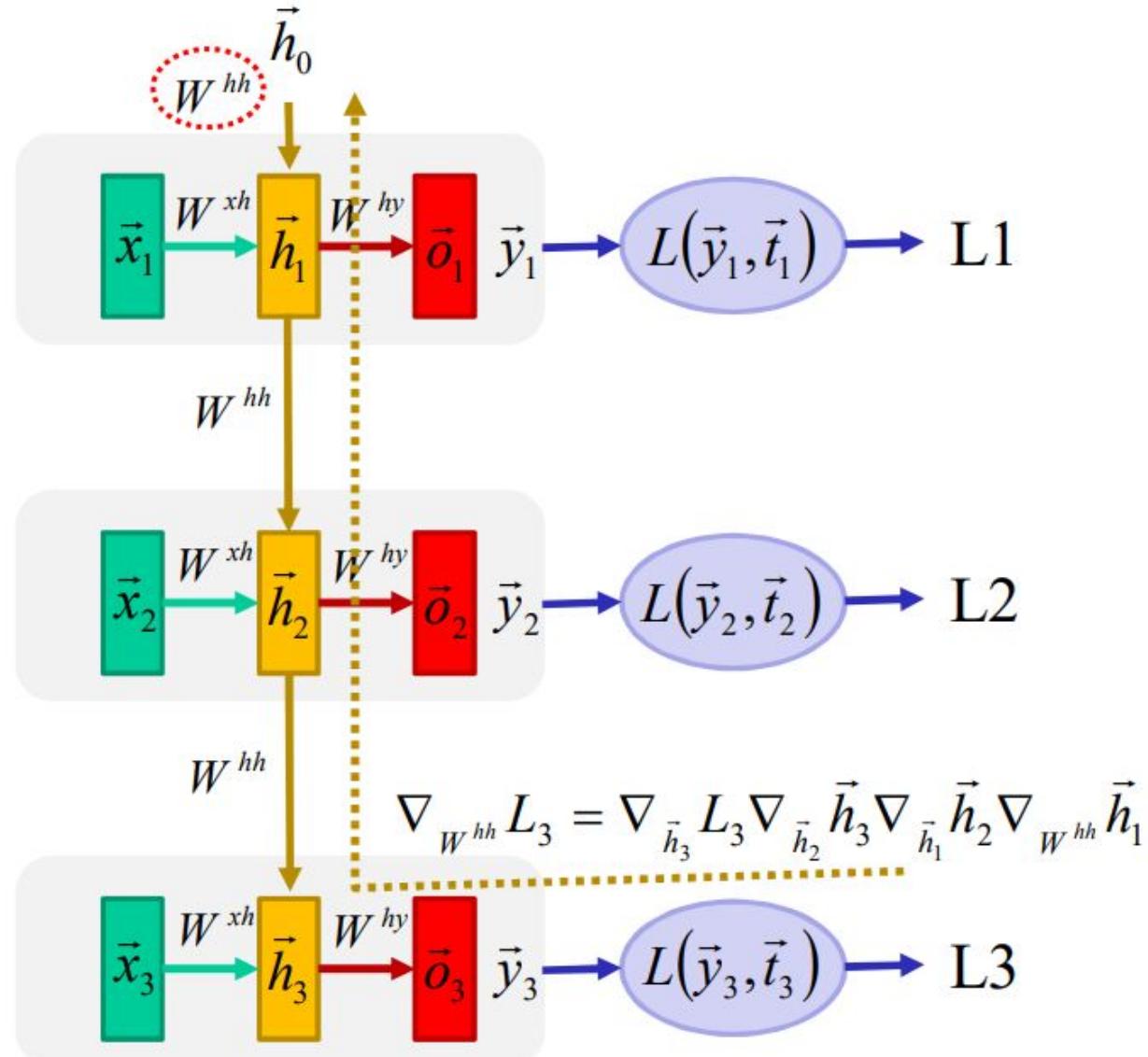
Réseau récurrent: gradient pour W^{hh}



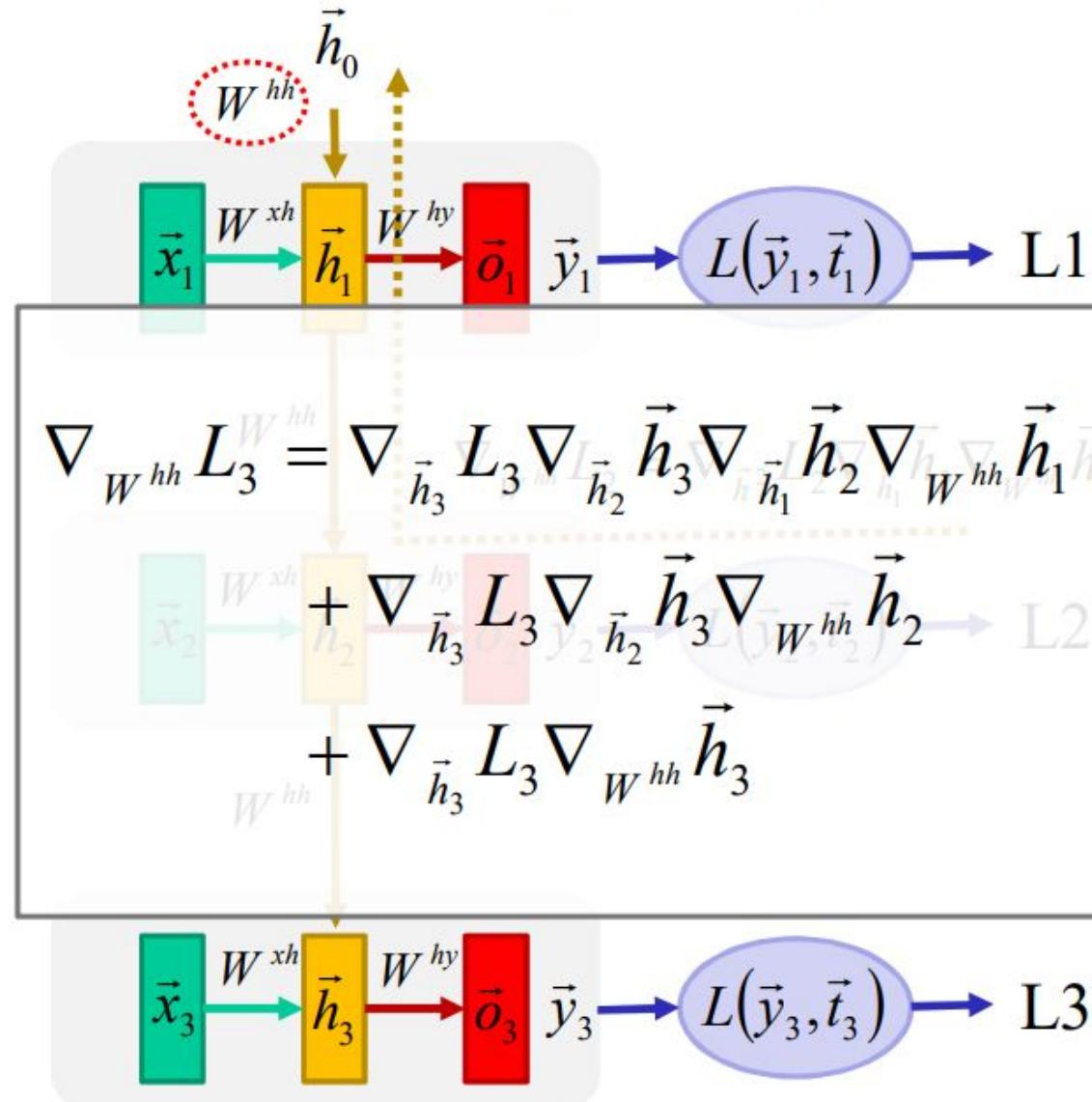
Réseau récurrent: gradient pour W^{hh}



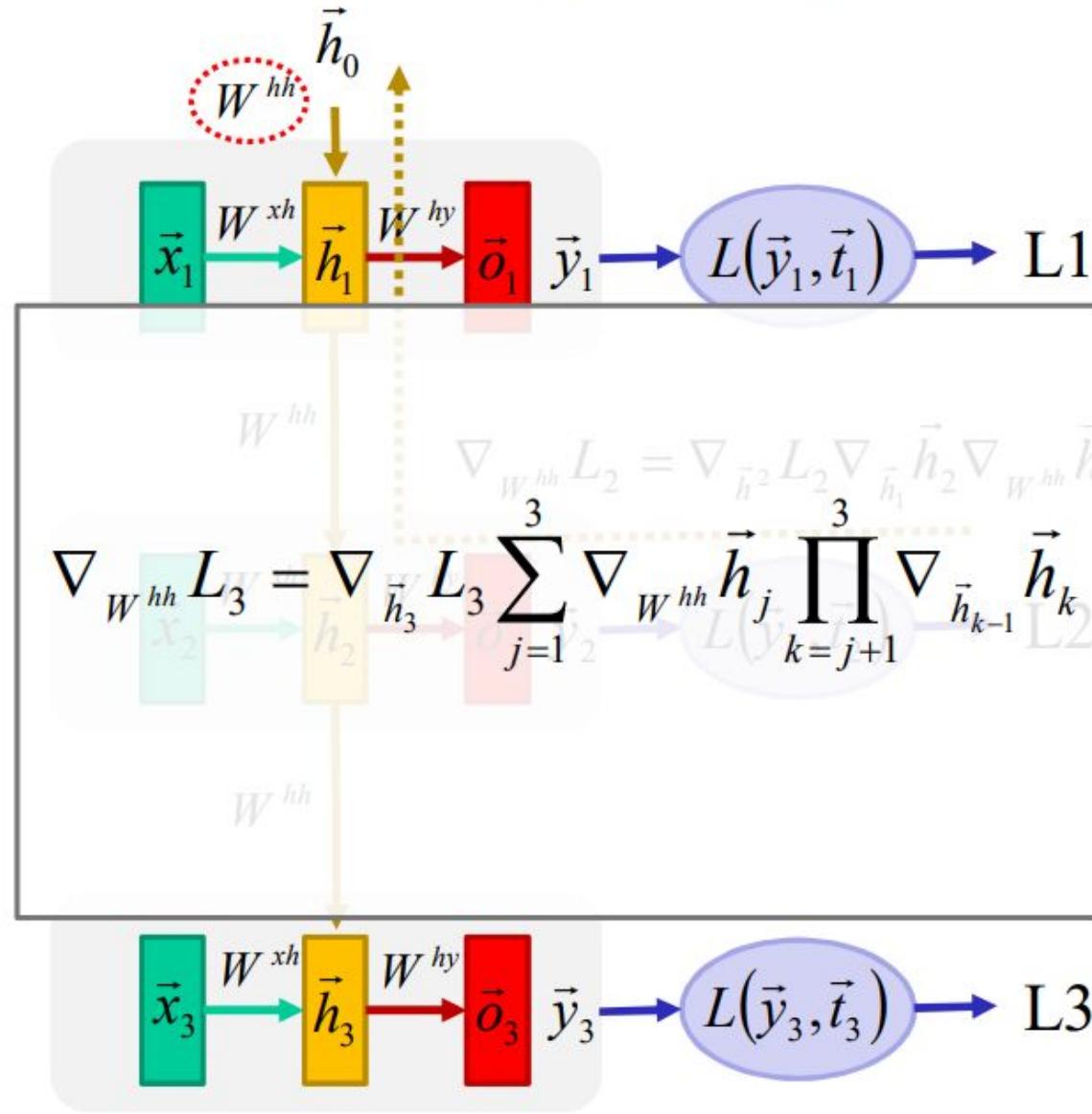
Réseau récurrent: gradient pour W^{hh}



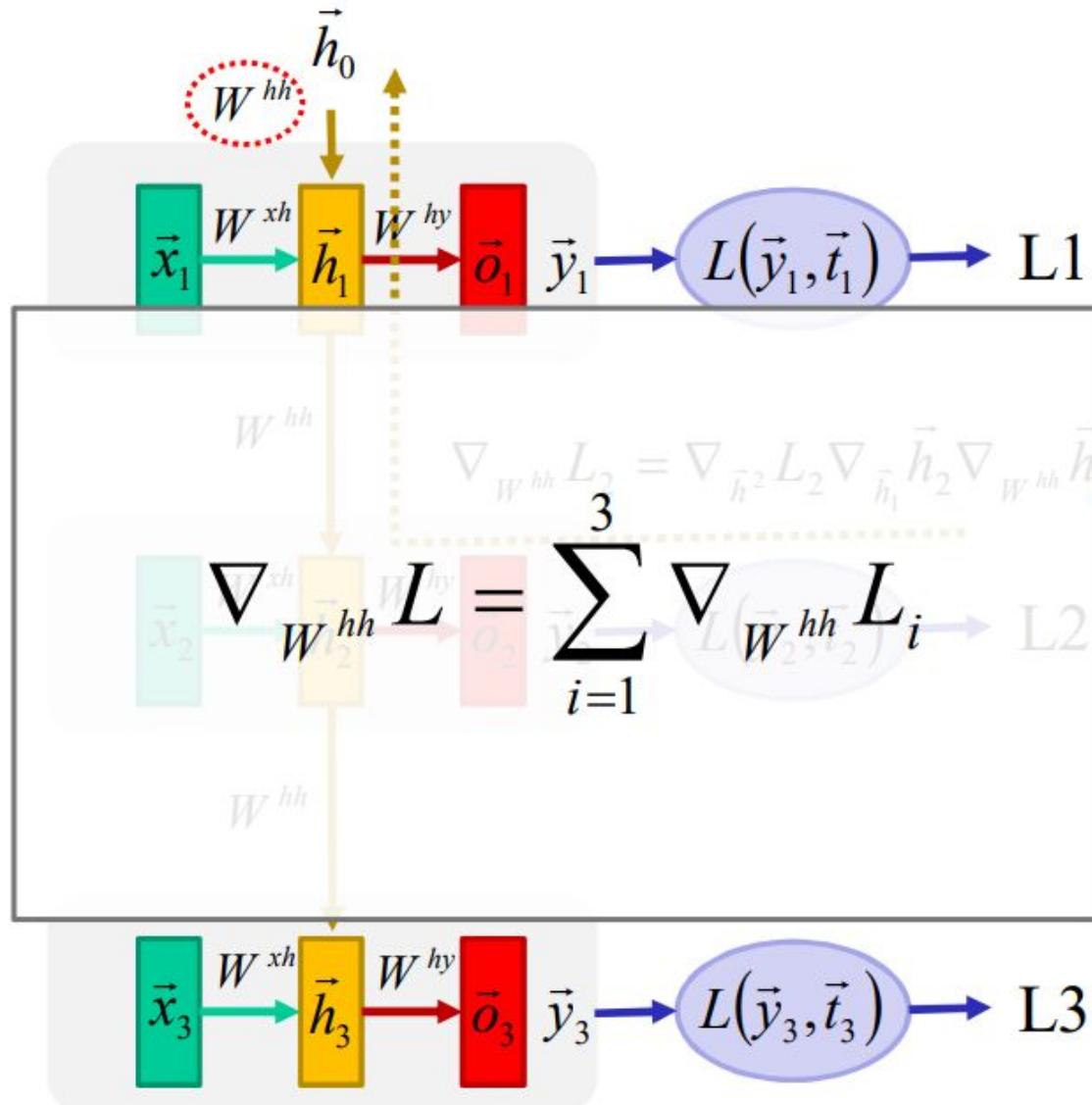
Réseau récurrent: gradient pour W^{hh}



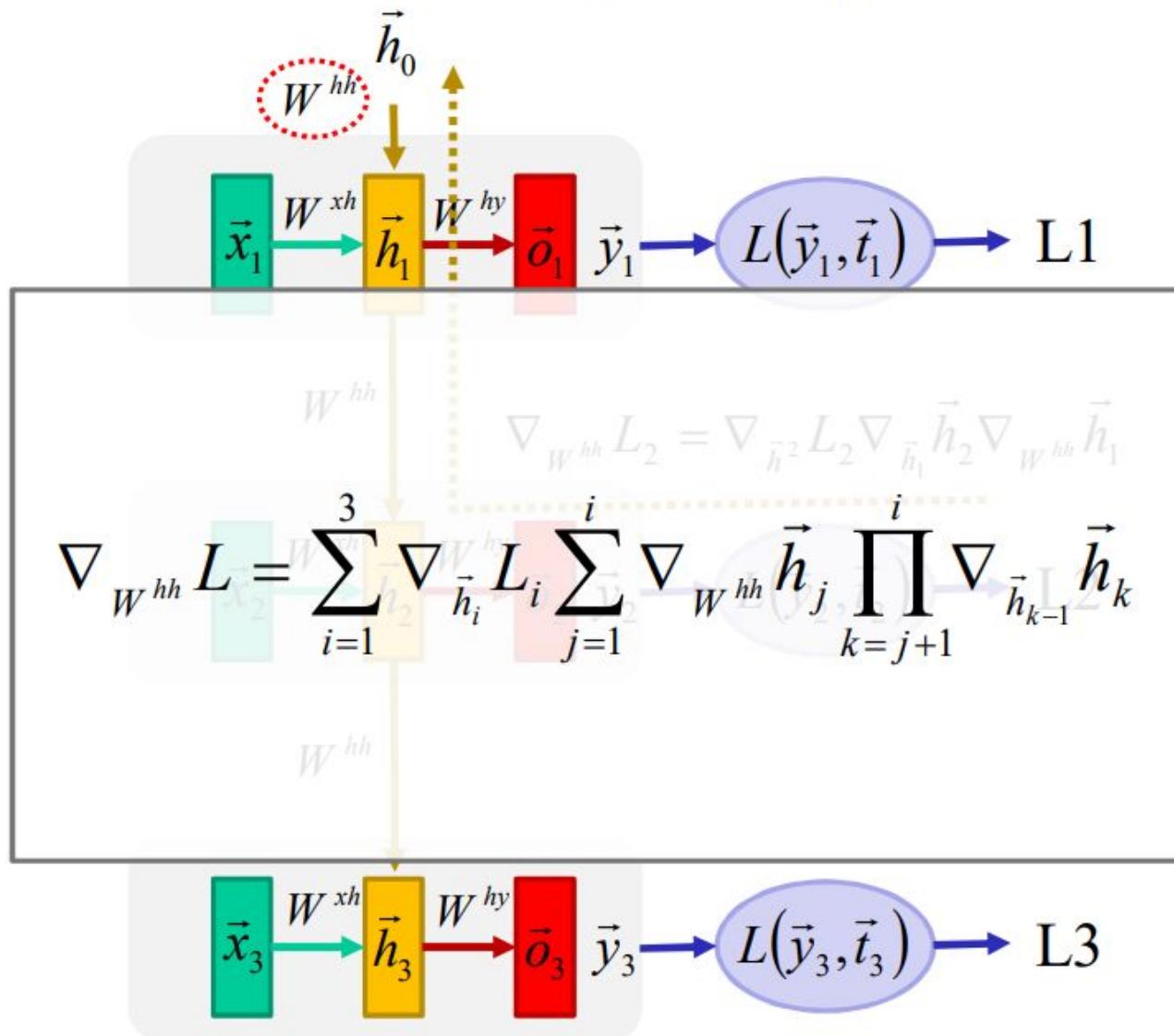
Réseau récurrent: gradient pour W^{hh}



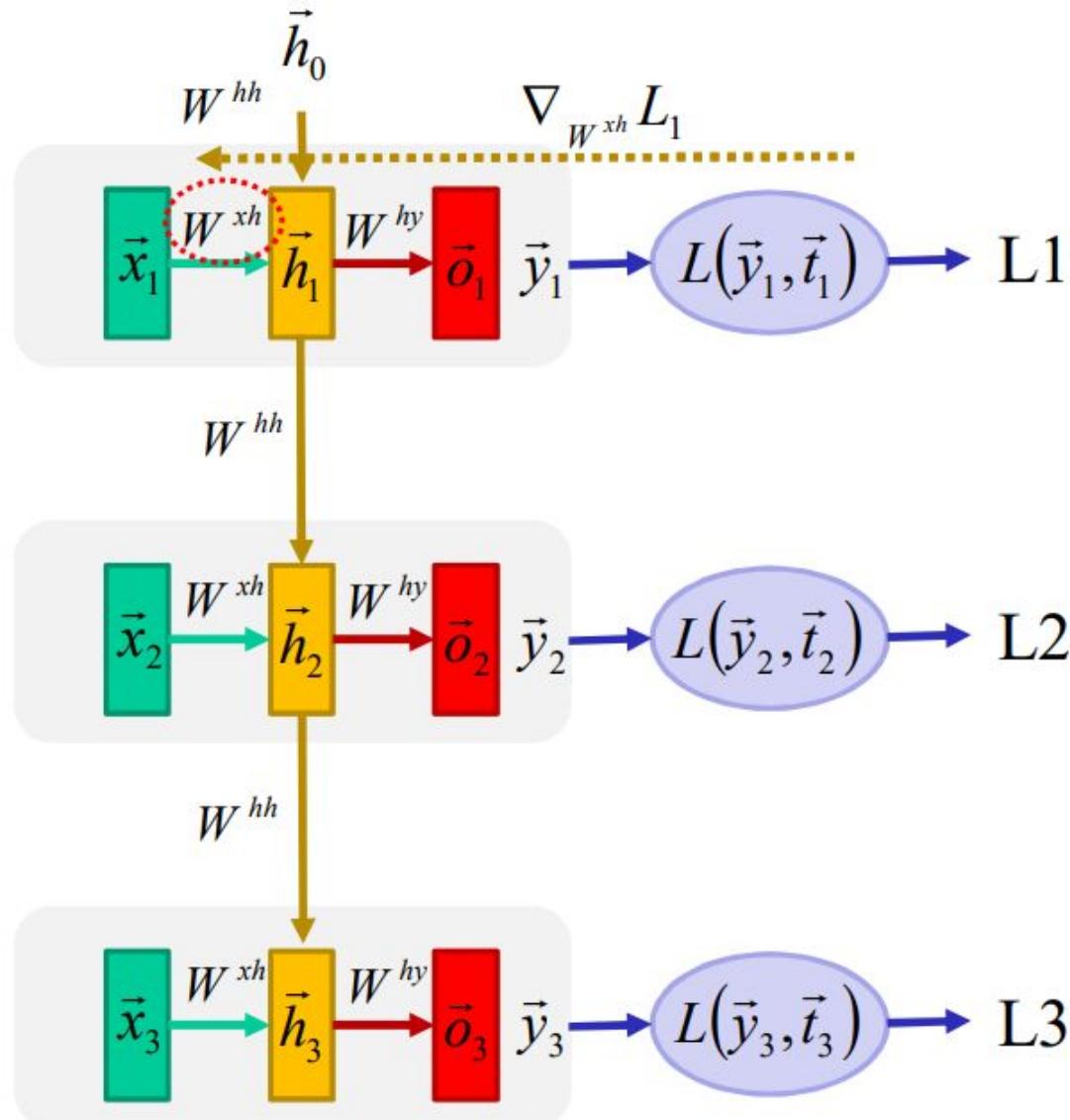
Réseau récurrent: gradient pour W^{hh}



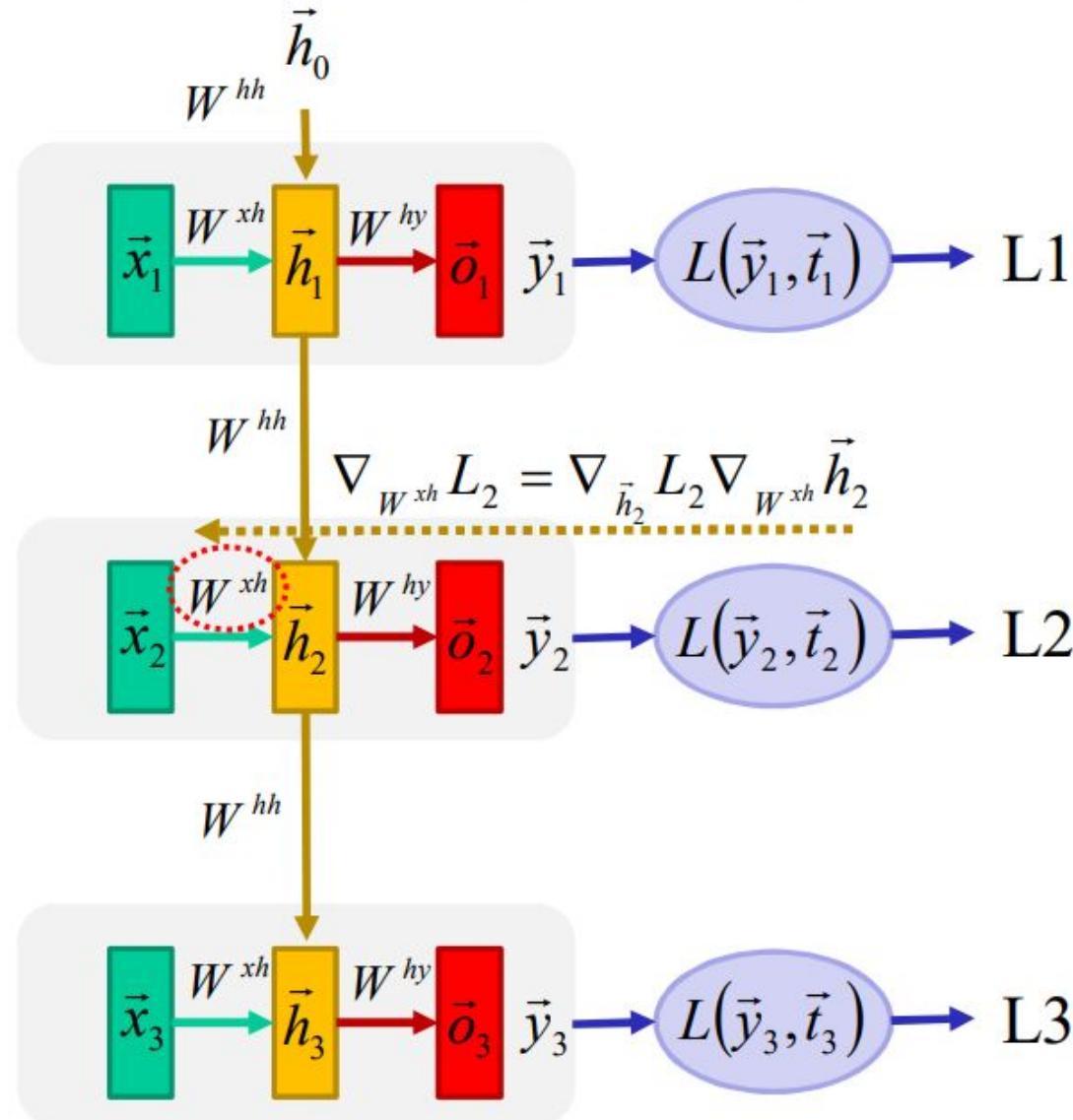
Réseau récurrent: gradient pour W^{hh}



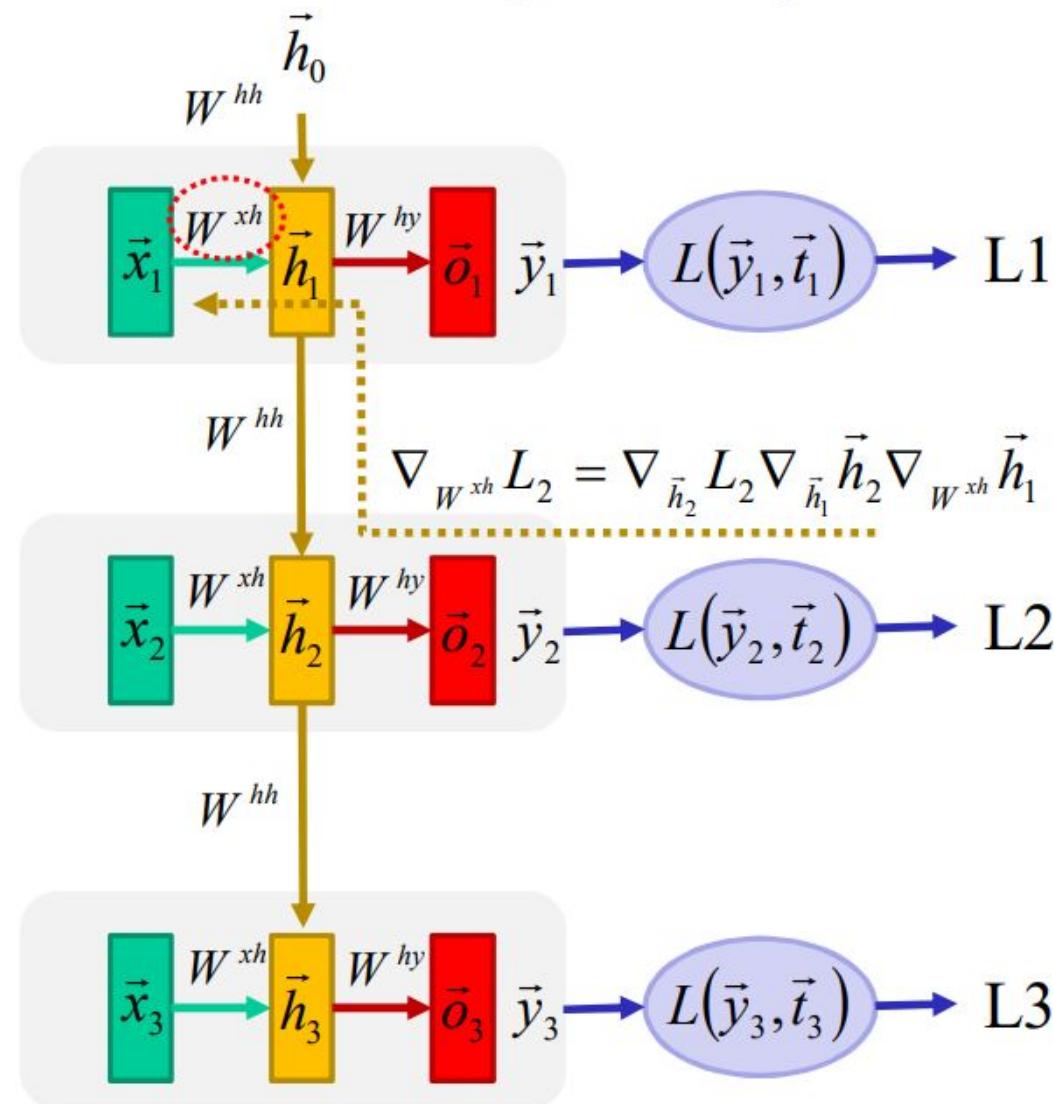
Réseau récurrent: gradient pour W^{xh}



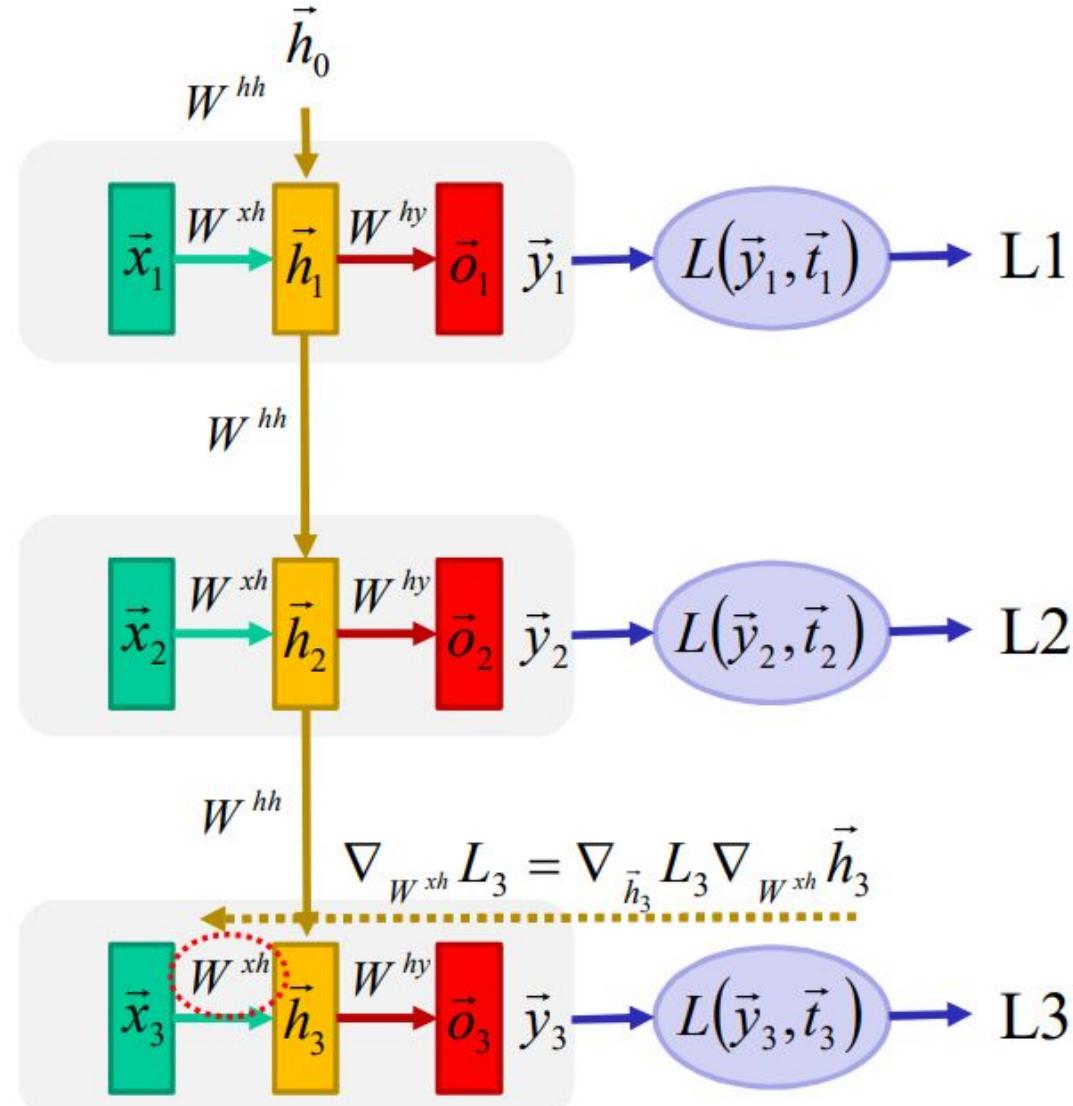
Réseau récurrent: gradient pour W^{xh}



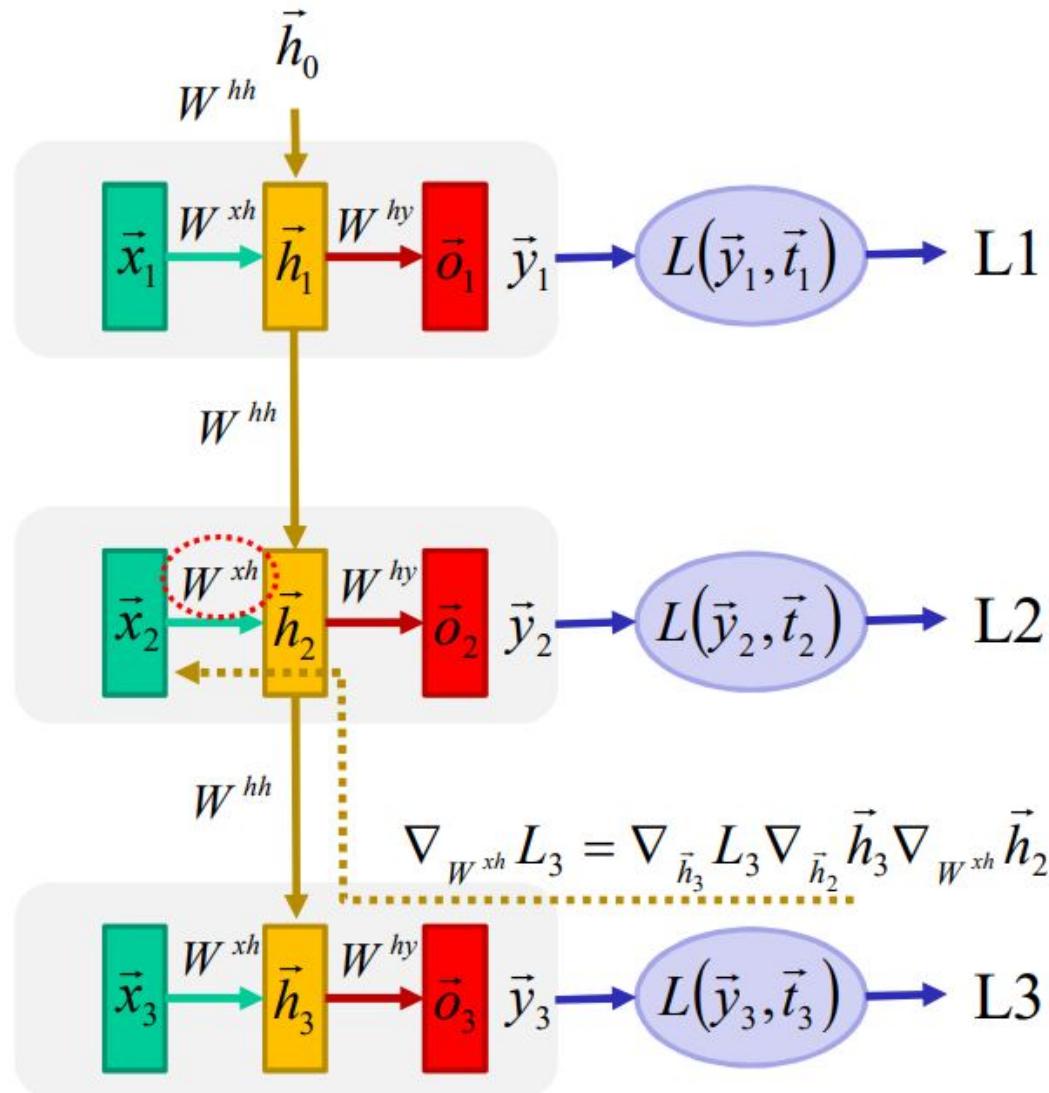
Réseau récurrent: gradient pour W^{xh}



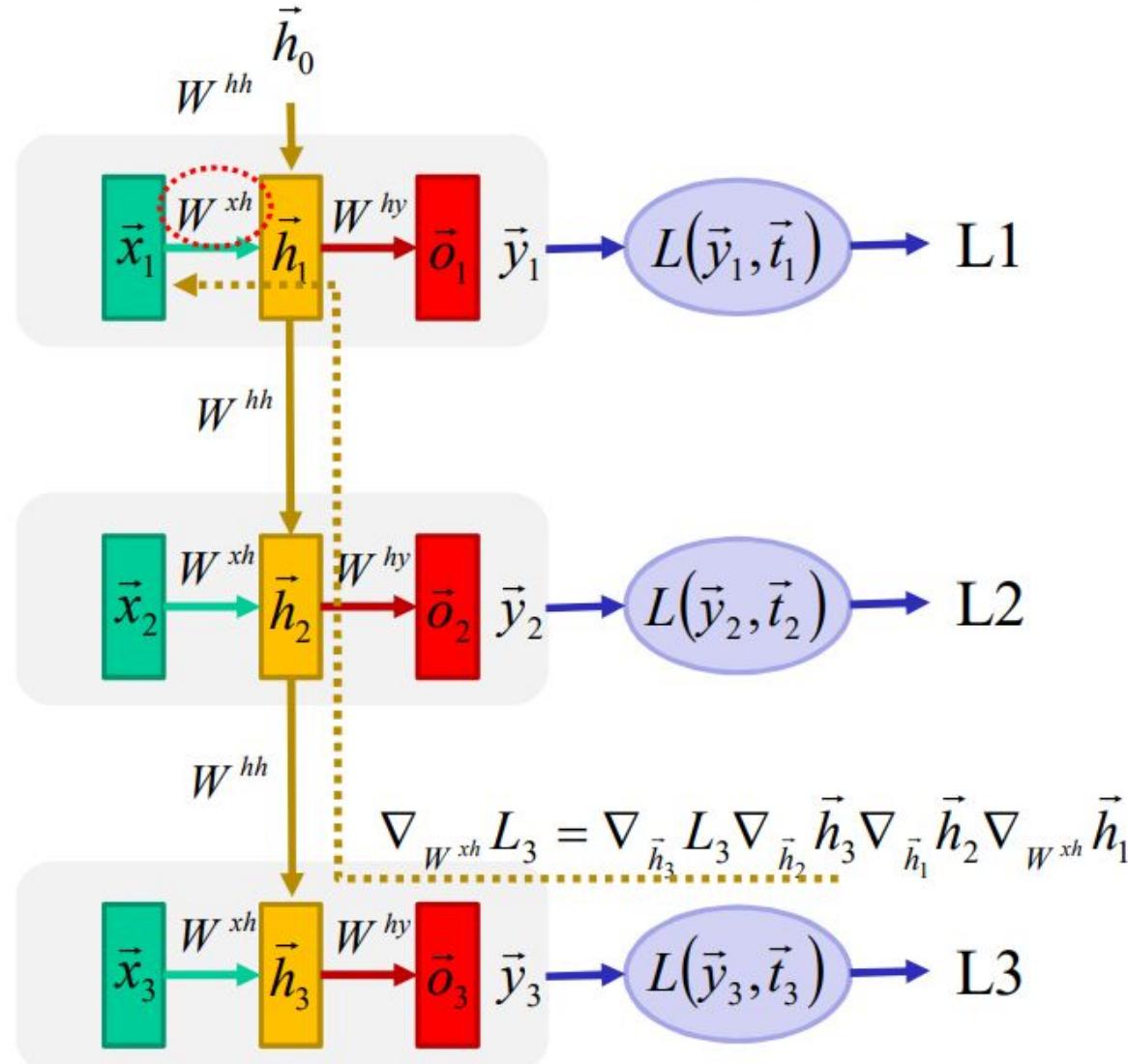
Réseau récurrent: gradient pour W^{xh}



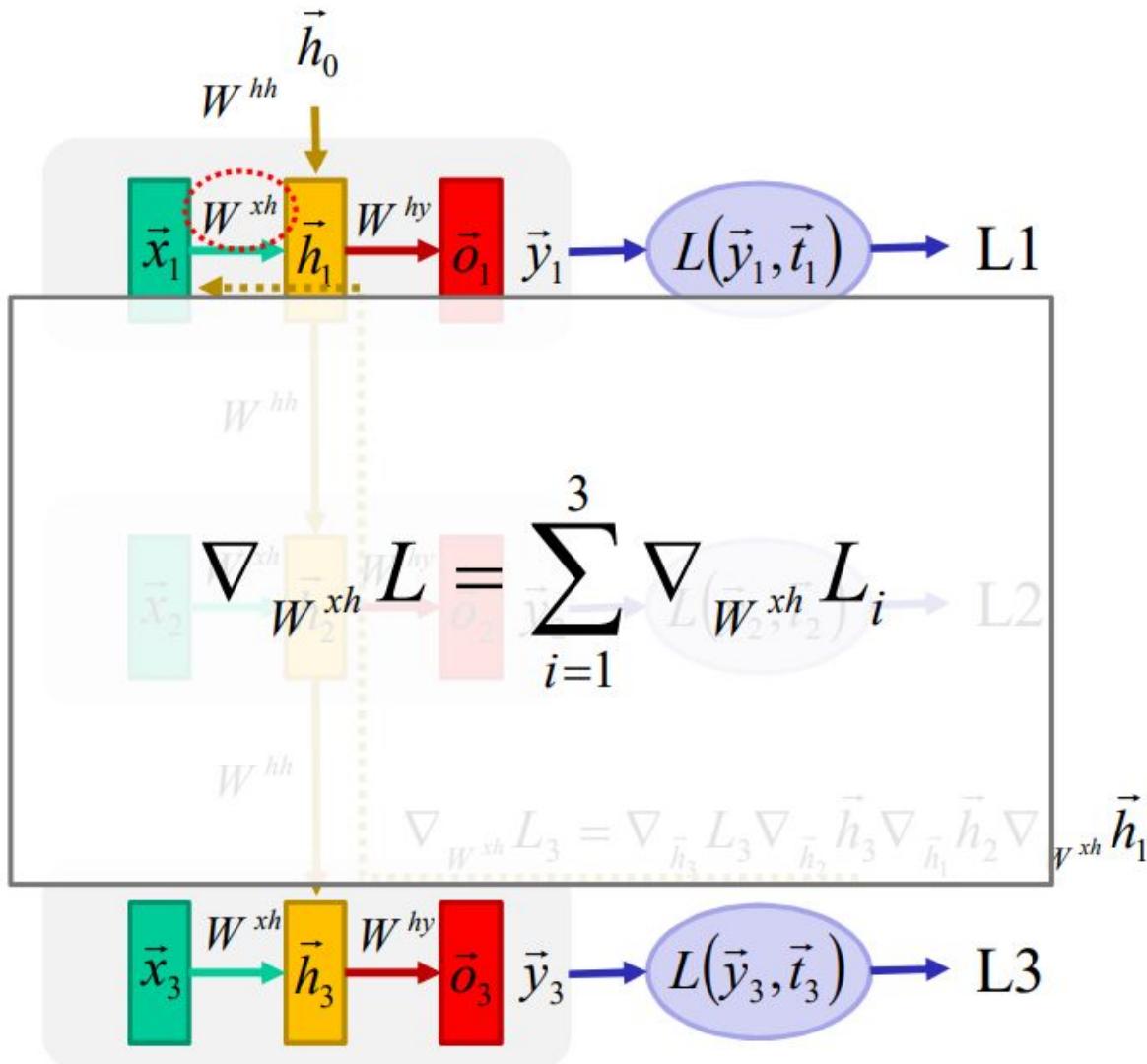
Réseau récurrent: gradient pour W^{xh}



Réseau récurrent: gradient pour W^{xh}



Réseau récurrent: gradient pour W^{xh}



Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

En réorganisant les termes:

$$\begin{aligned}\nabla_{W^{xh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot x_t \\ \nabla_{W^{hh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1}\end{aligned}$$

Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

En réorganisant les termes:

$$\nabla_{W^{xh}} L = \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t$$

$$\nabla_{W^{hh}} L = \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t$$

$$\begin{aligned} &= \sum_t^T \nabla_{h_t} L \cdot x_t \\ &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1} \end{aligned}$$

à définir

Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

En réorganisant les termes:

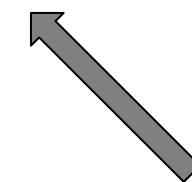
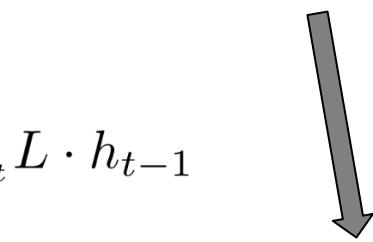
$$\begin{aligned}\nabla_{W^{xh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot x_t \\ \nabla_{W^{hh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1} \\ \nabla_{h_t} L &= \nabla_{h_{t+1}} L \cdot \nabla_{h_t} h_{t+1} + \nabla_{y_t} L \cdot \nabla_{h_t} y_t &= W^{hh} \nabla_{h_{t+1}} L + W^{hy} \nabla_{y_t} L\end{aligned}$$

Réseau récurrent: calcul du gradient

Semblaient complexes, mais ne l'est pas vraiment !

En réorganisant les termes:

$$\begin{aligned}\nabla_{W^{xh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot x_t && \text{facile à calculer} \\ \nabla_{W^{hh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1} \\ \nabla_{h_t} L &= \nabla_{h_{t+1}} L \cdot \nabla_{h_t} h_{t+1} + \nabla_{y_t} L \cdot \nabla_{h_t} y_t &= W^{hh} \nabla_{h_{t+1}} L + W^{hy} \nabla_{y_t} L\end{aligned}$$



récurativité,
oh no !

Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

En réorganisant les termes:

$$\begin{aligned}\nabla_{W^{xh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot x_t \\ \nabla_{W^{hh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1} \\ \nabla_{h_t} L &= \nabla_{h_{t+1}} L \cdot \nabla_{h_t} h_{t+1} + \nabla_{y_t} L \cdot \nabla_{h_t} y_t &= W^{hh} \nabla_{h_{t+1}} L + W^{hy} \nabla_{y_t} L \\ \nabla_{h_T} L &= \nabla_{y_T} L \cdot \nabla_{h_T} y_T &= W^{hy} \nabla_{y_T} L\end{aligned}$$

Réseau récurrent: calcul du gradient

Semblé complexe, mais ne l'est pas vraiment !

En réorganisant les termes:

$$\begin{aligned}\nabla_{W^{xh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{xh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot x_t \\ \nabla_{W^{hh}} L &= \sum_t^T \nabla_{h_t} L \cdot \nabla_{W^{hh}} h_t &= \sum_t^T \nabla_{h_t} L \cdot h_{t-1} \\ \nabla_{h_t} L &= \nabla_{h_{t+1}} L \cdot \nabla_{h_t} h_{t+1} + \nabla_{y_t} L \cdot \nabla_{h_t} y_t &= W^{hh} \nabla_{h_{t+1}} L + W^{hy} \nabla_{y_t} L \\ \nabla_{h_T} L &= \nabla_{y_T} L \cdot \nabla_{h_T} y_T &= W^{hy} \nabla_{y_T} L\end{aligned}$$

On peut donc partir de la fin T et calculer des sommes courantes !

Cas de base
facile à
calculer !

Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

```
44     # backward pass: compute gradients going backwards
45     dwxh, dwhh, dwhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
46     dbh, dby = np.zeros_like(bh), np.zeros_like(by)
47     dhnext = np.zeros_like(hs[0])
48     for t in reversed(xrange(len(inputs))):
49         dy = np.copy(ps[t])
50         dy[targets[t]] -= 1 # backprop into y. see http://cs231n.github.io/neural-networks-case-study/#grad if confused here
51         dwhy += np.dot(dy, hs[t].T)
52         dby += dy
53         dh = np.dot(Why.T, dy) + dhnext # backprop into h
54         ddraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
55         dbh += ddraw
56         dwxh += np.dot(ddraw, xs[t].T)
57         dwhh += np.dot(ddraw, hs[t-1].T)
58         dhnext = np.dot(Whh.T, ddraw)
59     for dparam in [dwxh, dwhh, dwhy, dbh, dby]:
60         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61     return loss, dwxh, dwhh, dwhy, dbh, dby, hs[len(inputs)-1]
```

Pas besoin de recalculer tous les termes intermédiaires pour chaque *timestep*

Voir https://d2l.ai/chapter_recurrent-neural-networks/bptt.html pour plus d'informations

Réseau récurrent: calcul du gradient

Semble complexe, mais ne l'est pas vraiment !

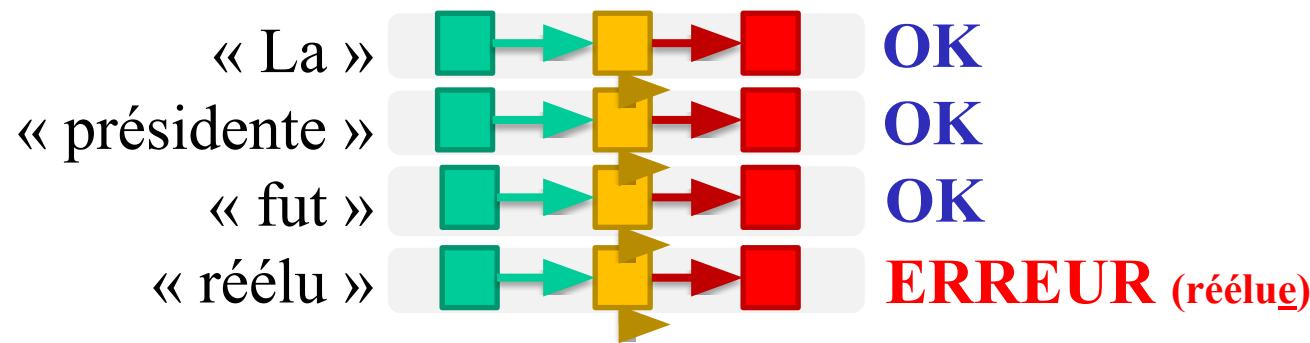
```
44     # backward pass: compute gradients going backwards
45     dwxh, dwhh, dwhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
46     dbh, dby = np.zeros_like(bh), np.zeros_like(by)
47     dhnext = np.zeros_like(hs[0])
48     for t in reversed(xrange(len(inputs))):  
49         dy = np.copy(ps[t])
50         dy[targets[t]] -= 1 # backprop into y. see http://cs231n.github.io/neural-networks-case-study/#grad if confused here
51         dWhy += np.dot(dy, hs[t].T)
52         dby += dy
53         dh = np.dot(Why.T, dy) + dhnext # backprop into h
54         ddraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
55         dbh += ddraw
56         dwxh += np.dot(ddraw, xs[t].T)
57         dwhh += np.dot(ddraw, hs[t-1].T)
58         dhnext = np.dot(Whh.T, ddraw)
59     for dparam in [dwxh, dwhh, dwhy, dbh, dby]:
60         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61     return loss, dwxh, dwhh, dwhy, dbh, dby, hs[len(inputs)-1]
```

Pas besoin de recalculer tous les termes intermédiaires pour chaque *timestep*

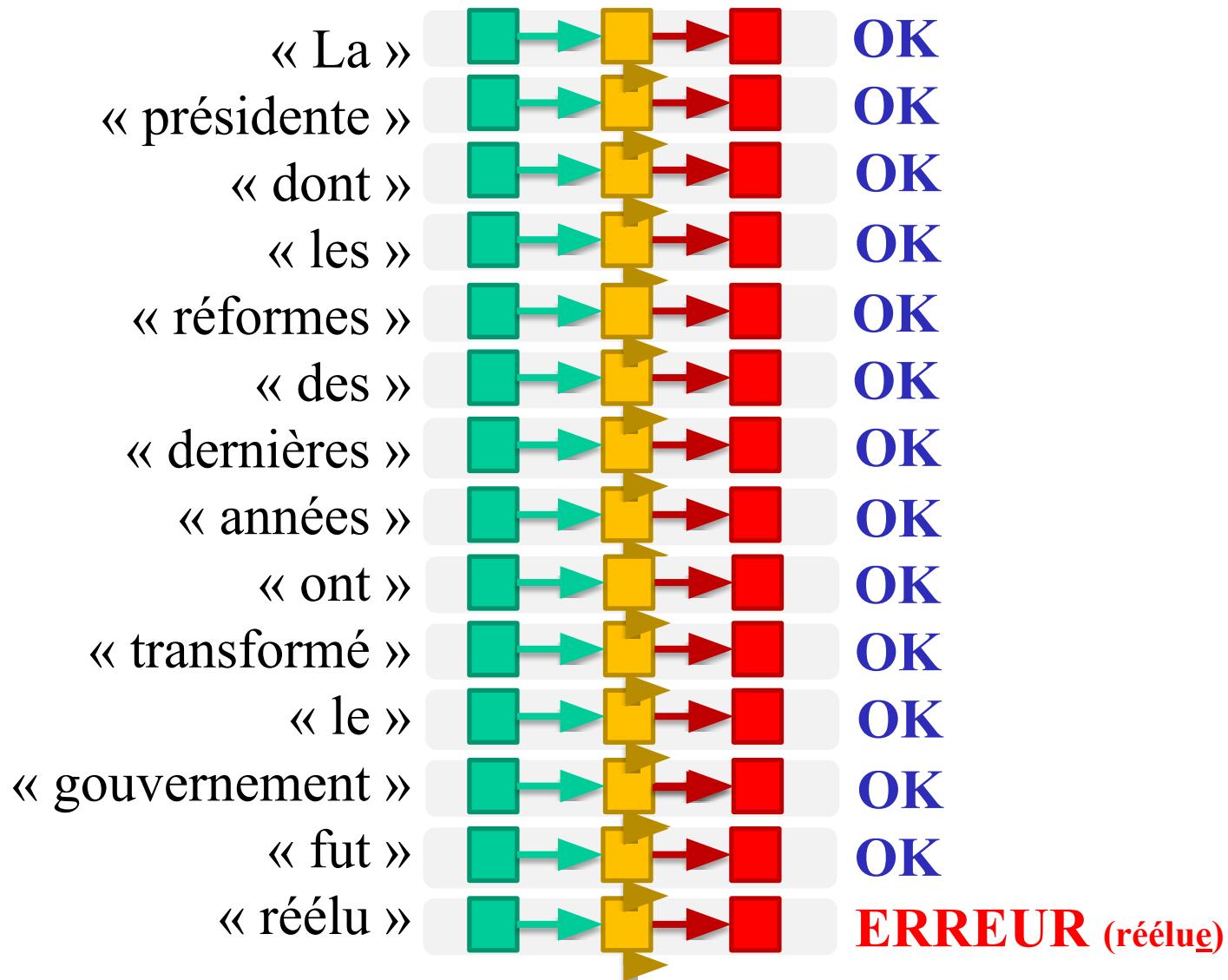
Voir https://d2l.ai/chapter_recurrent-neural-networks/bptt.html pour plus d'informations

Exemples: analyse grammaticale

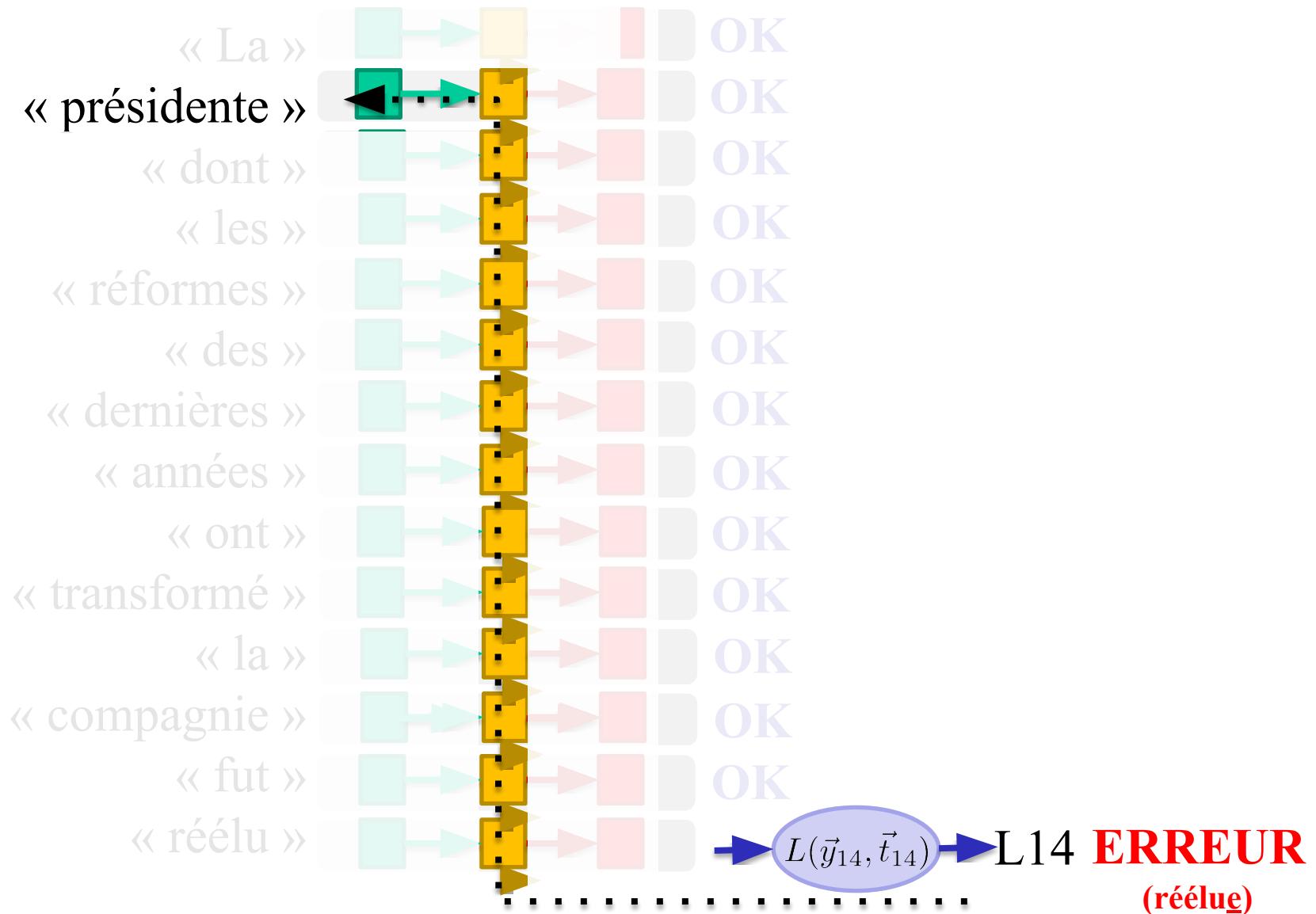
Entraîner un réseau à détecter des erreurs grammaticales

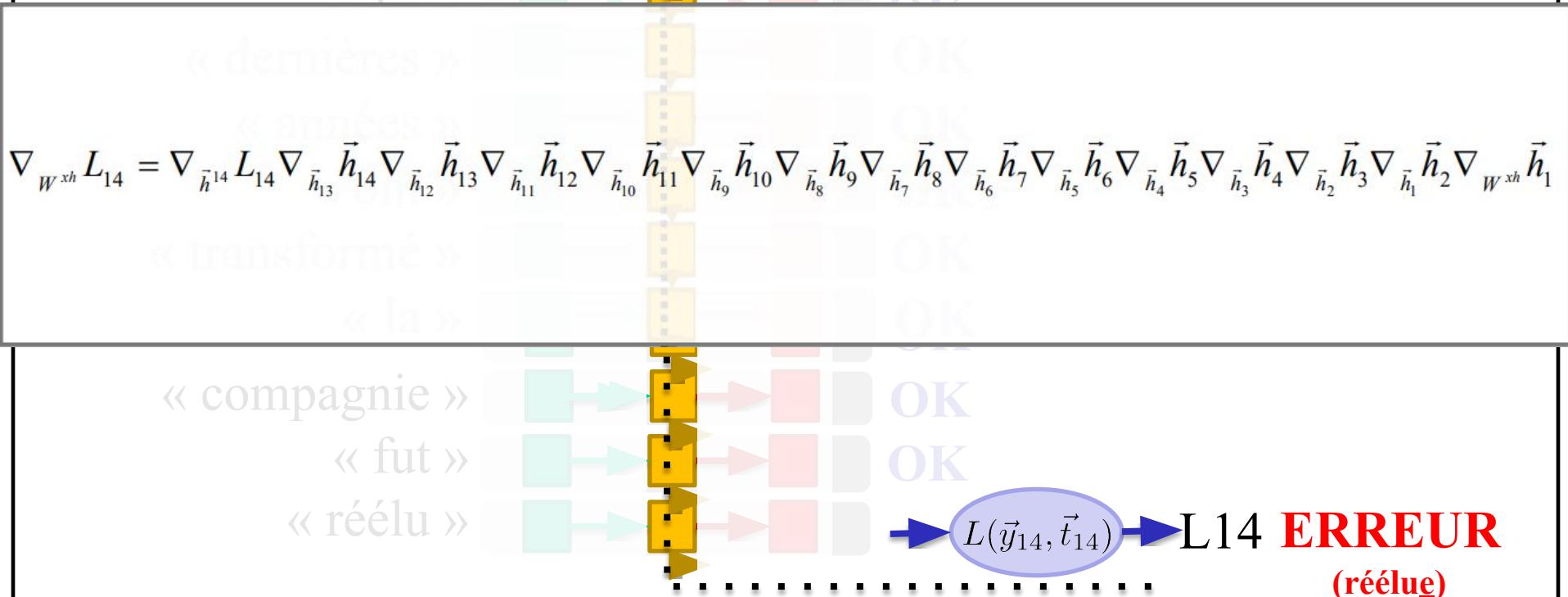
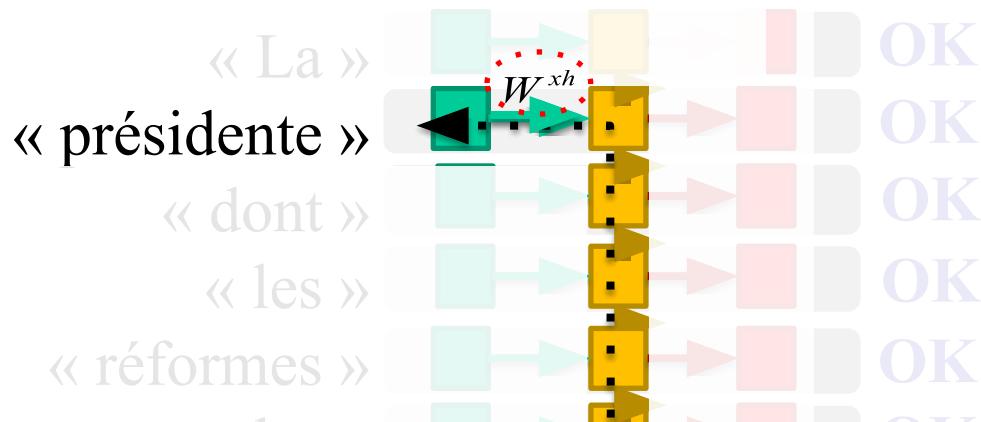


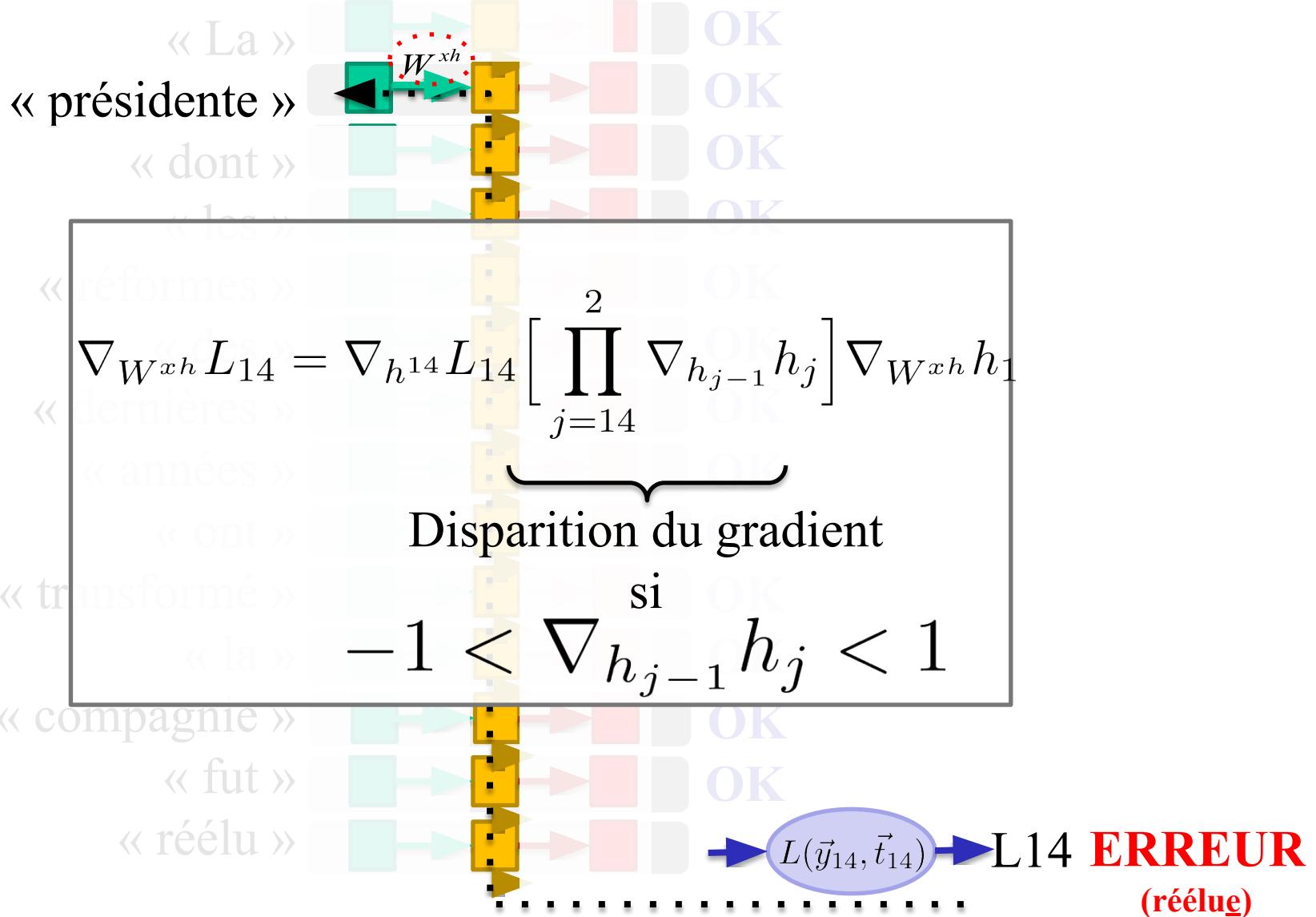
Exemple d'une relation à **courte distance**
(1 mot sépare « présidente » de « réélu »)

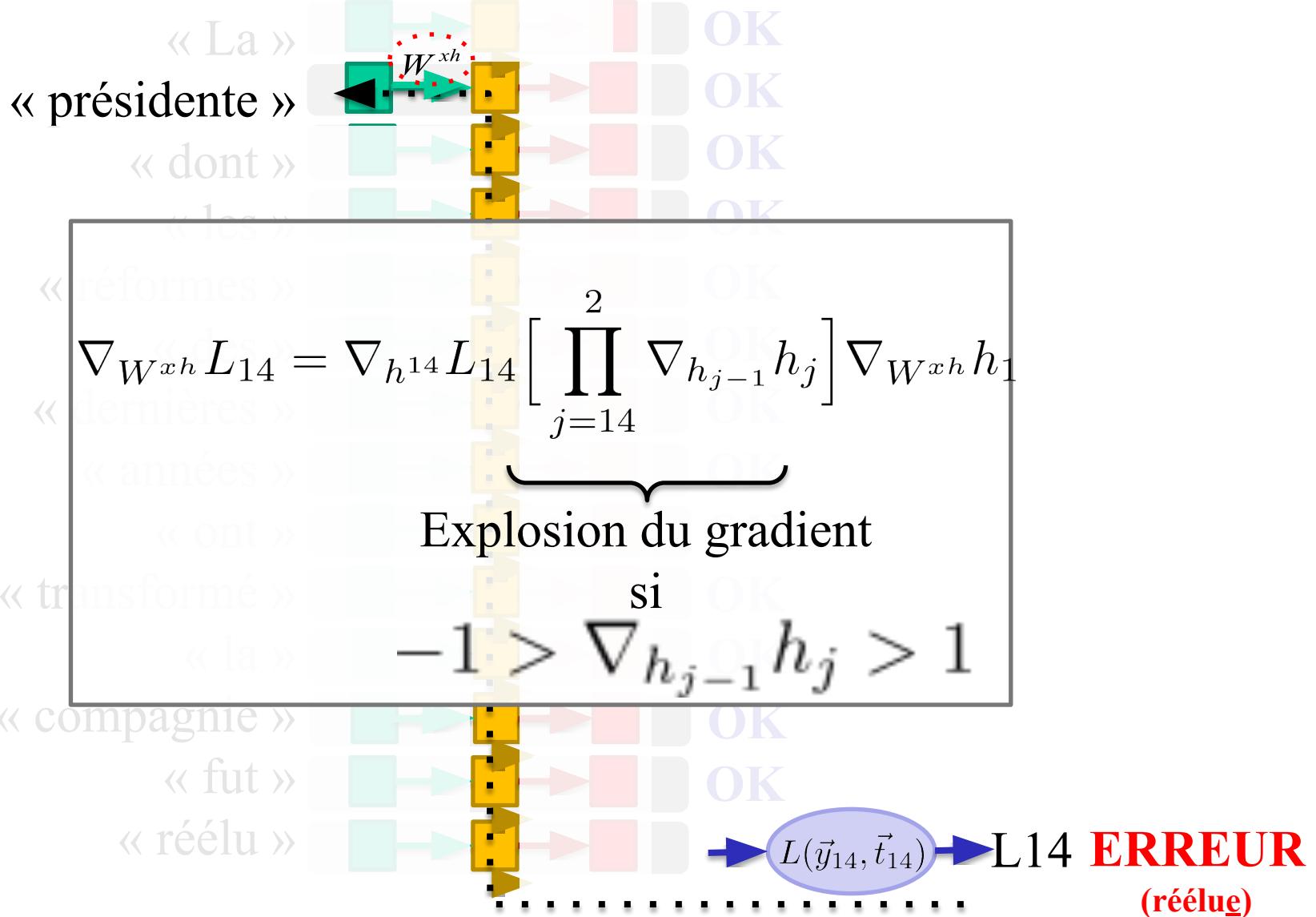


Exemple d'une relation à **longue distance**
 (12 mots séparent « présidente » de « réélu »)

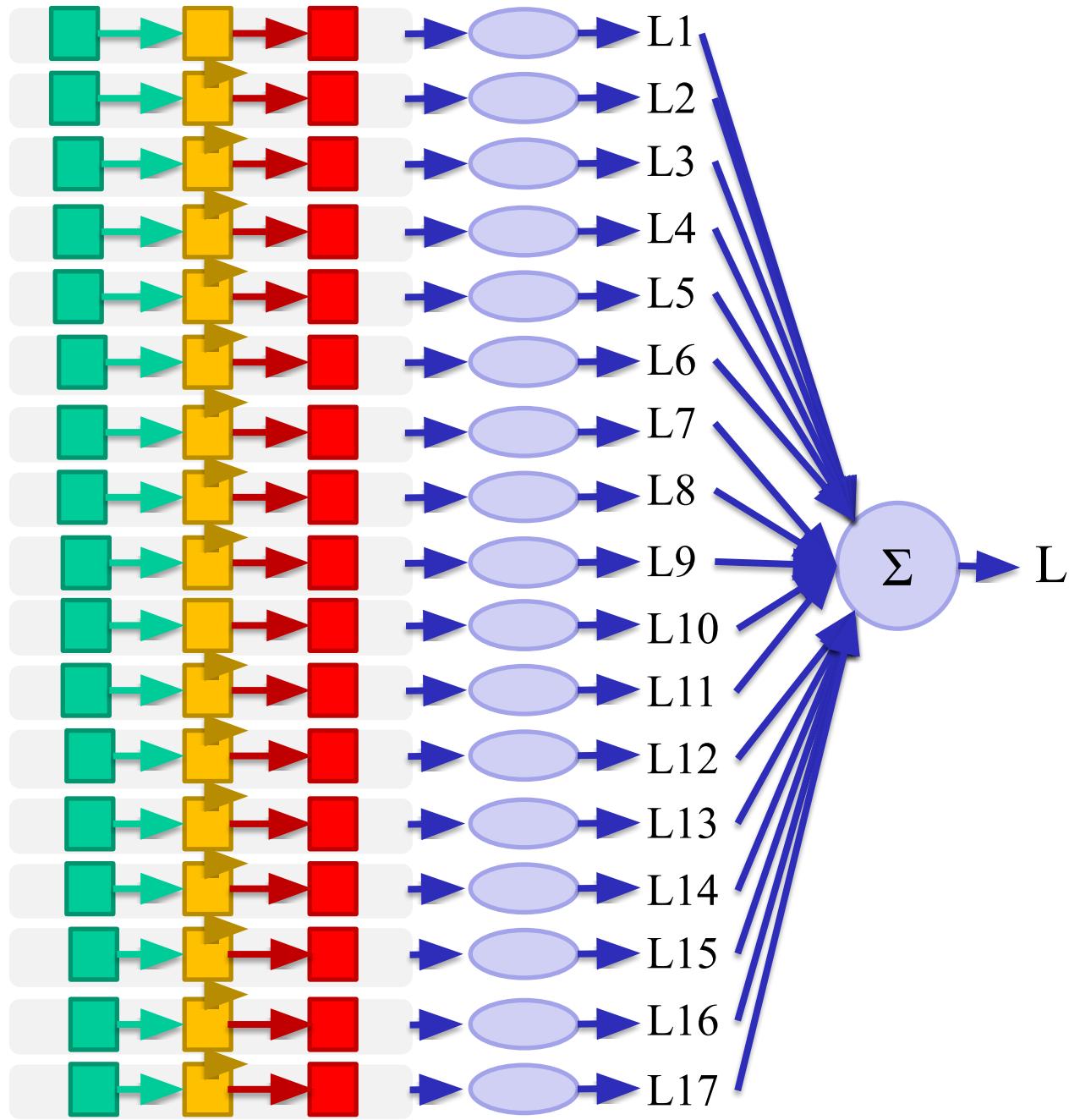




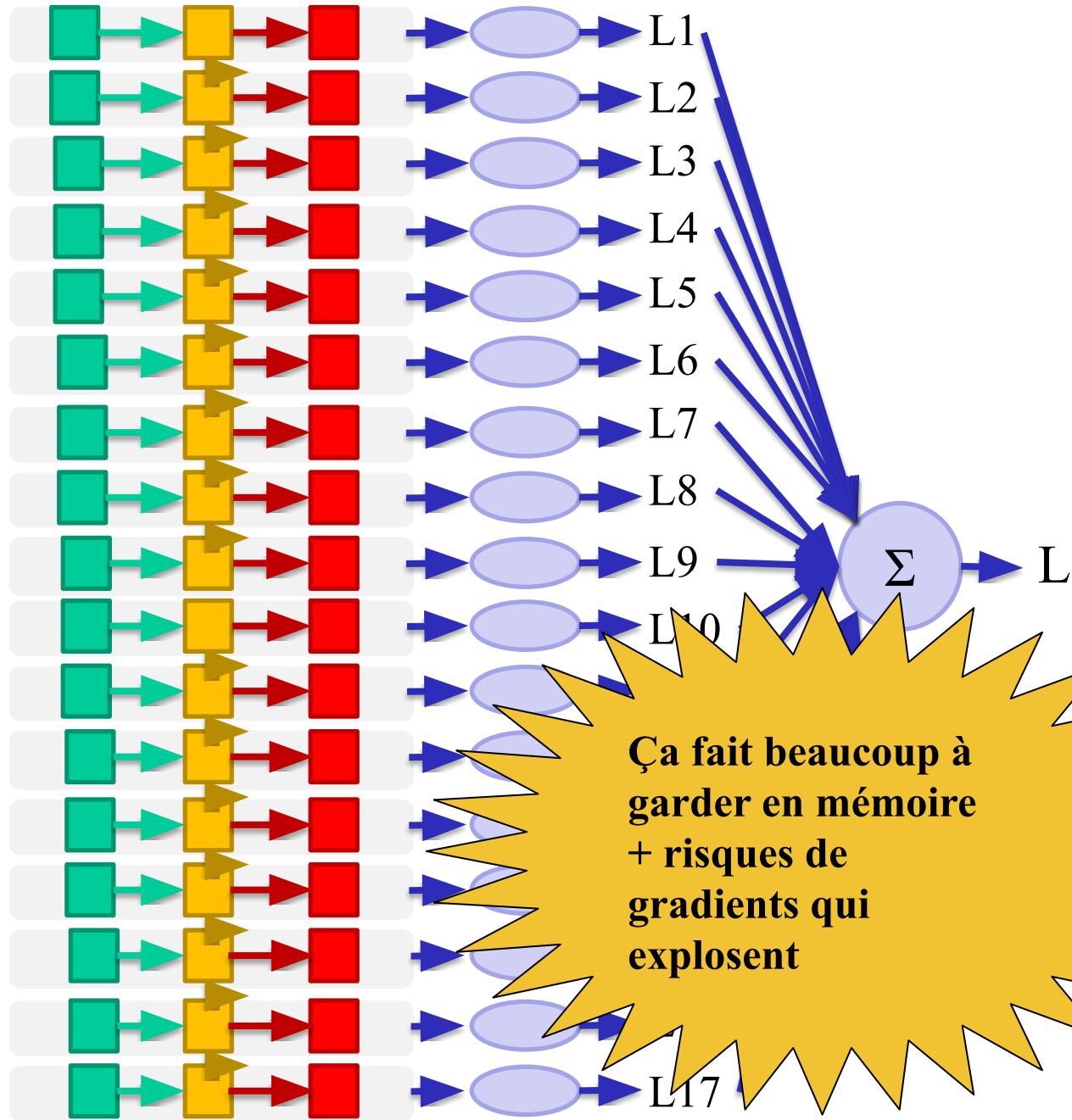




Propagation avant
Rétro-propagation



Propagation avant
Rétro-propagation



Ça fait beaucoup à
garder en mémoire
+ risques de
gradients qui
explosent

Les réseaux récurrents ont un inconvenient majeur:

difficile à établir des
relations à longue distance

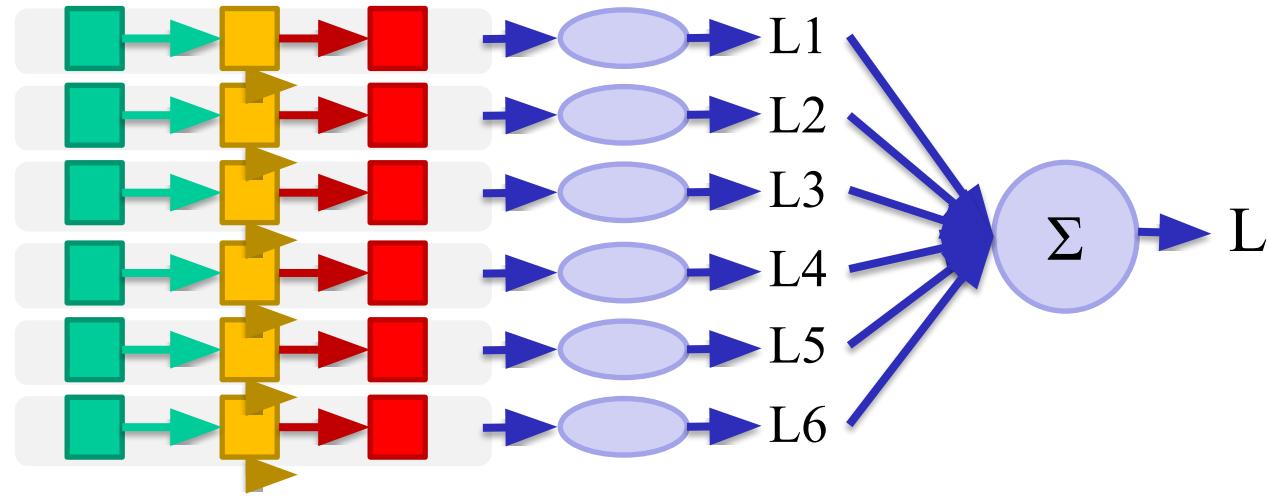
Solutions:

Rétropropagation tronquée

Gated recurrent unit : GRU

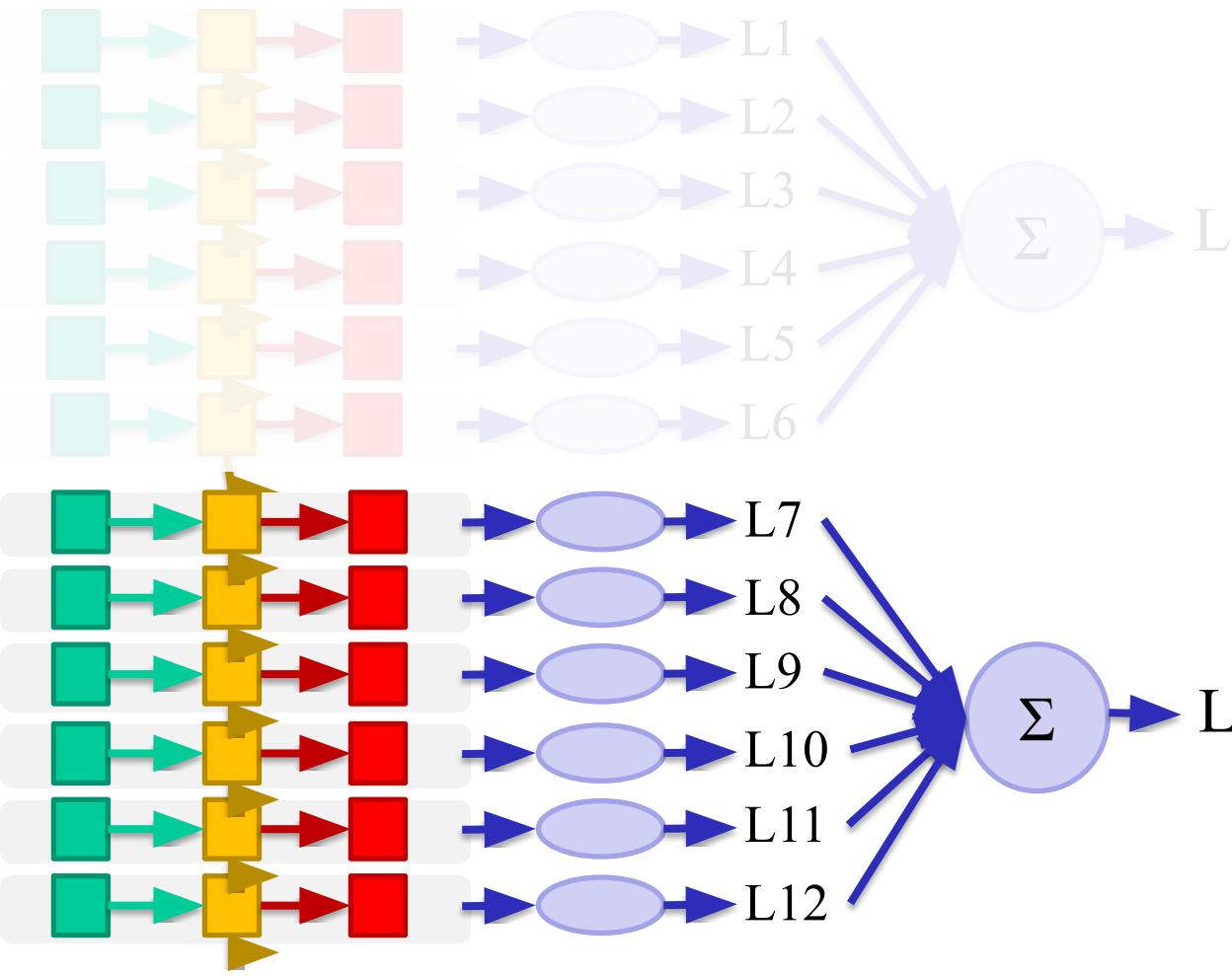
Long-Short Term Memory : LSTM

Propagation avant
↑
↓ Rétro-propagation



Lorsque les séquences sont trop longues
On entraîne le réseau par fenêtres coulissantes

Propagation avant
Rétro-propagation



on propage le contenu de la couche cachée
d'une section à l'autre.

Propagation avant
Rétro-propagation

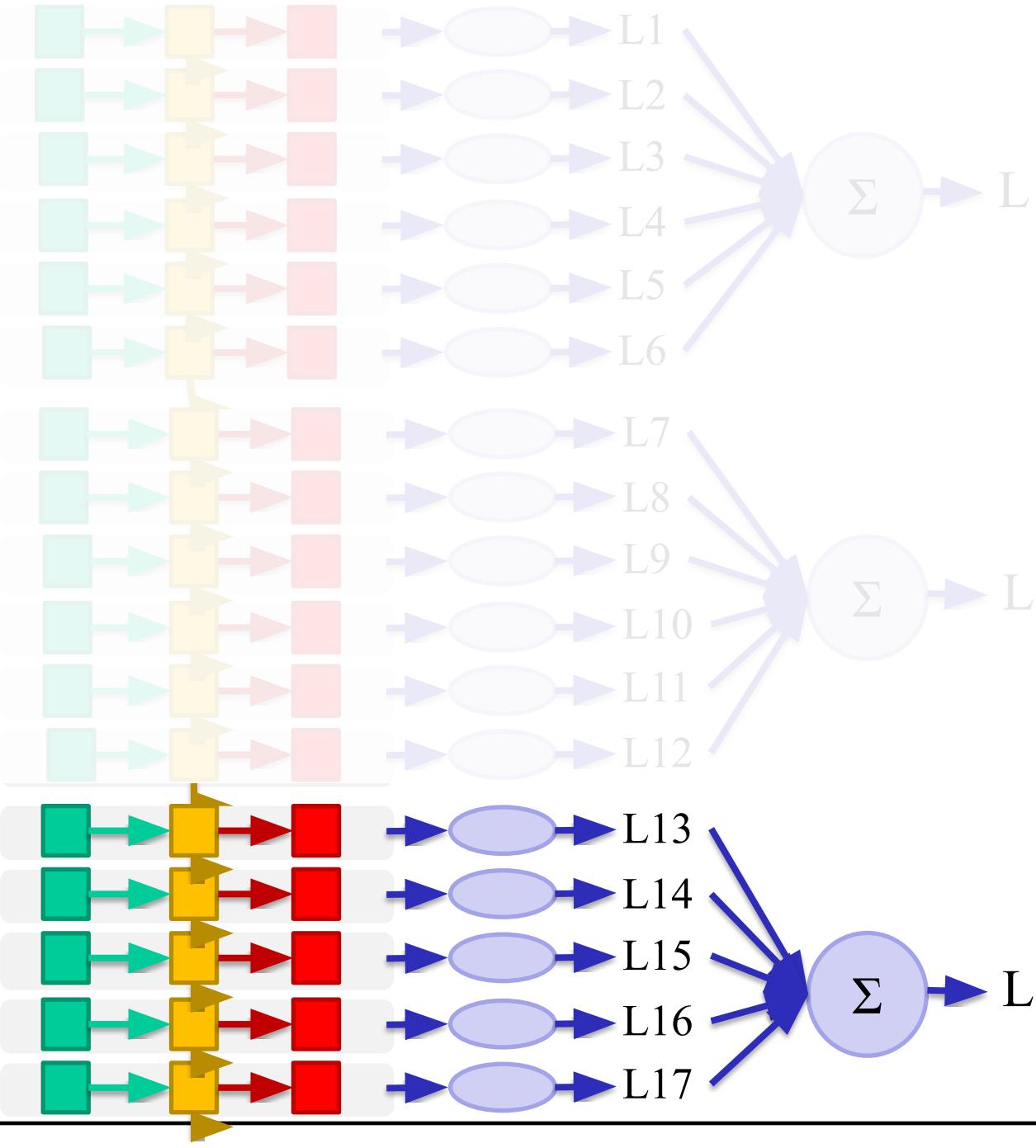
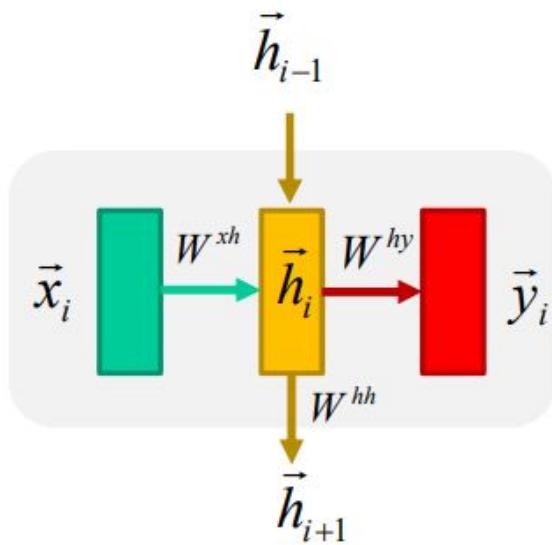
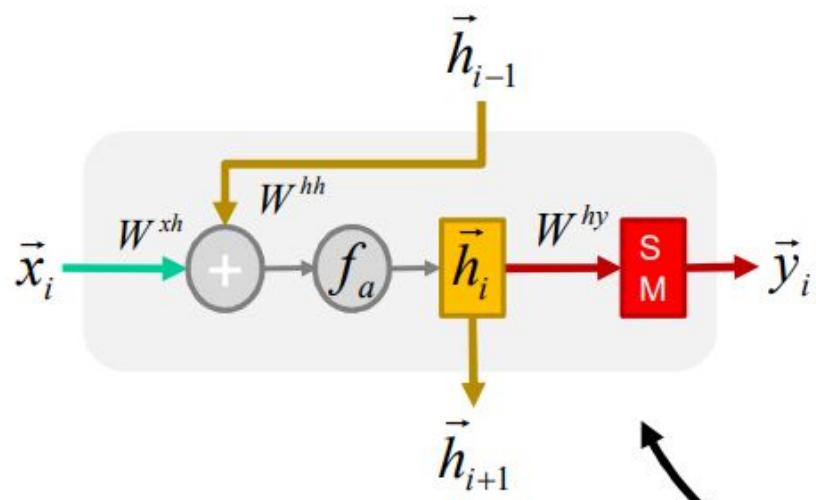


Illustration + formulation d'un RNN



$$\begin{aligned}\vec{h}_i &= f_a \left(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1} \right) \\ \hat{y}_i &= W^{hy} \vec{h}_i \\ \vec{y}_i &= SMAX(\hat{y}_i)\end{aligned}$$

Autre illustration du même RNN



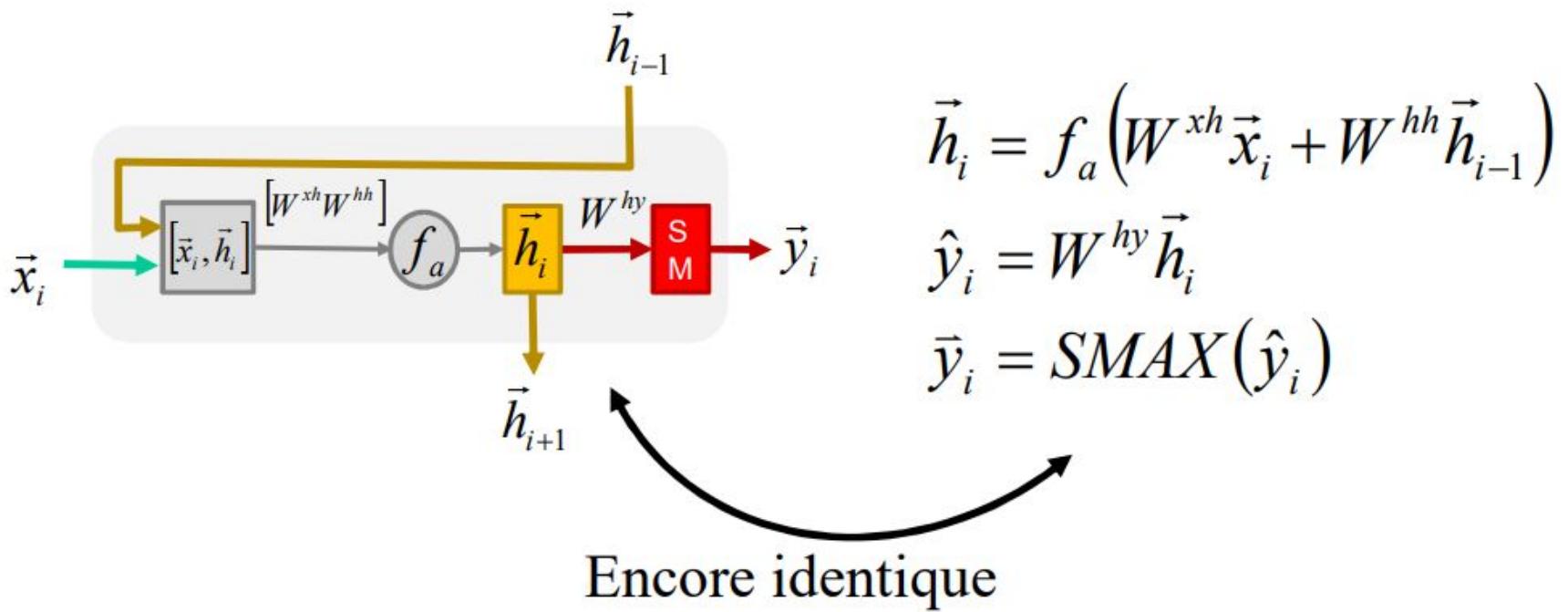
$$\vec{h}_i = f_a(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

$$\hat{y}_i = W^{hy}\vec{h}_i$$

$$\bar{y}_i = \text{SMAX}(\hat{y}_i)$$

Identique

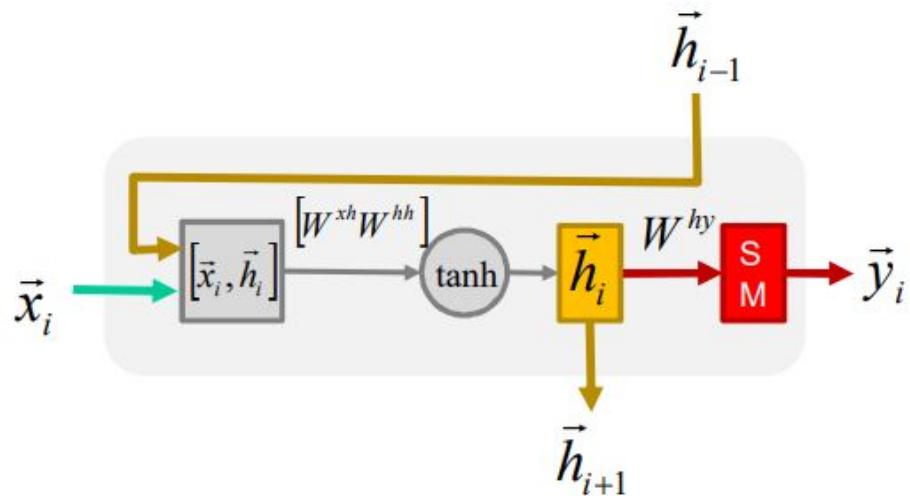
Autre illustration du même RNN



GRU (Gated Recurrent Unit)

Modif 1

$$f_a = \tanh$$



$$\vec{h}_i = \tanh\left(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}\right)$$

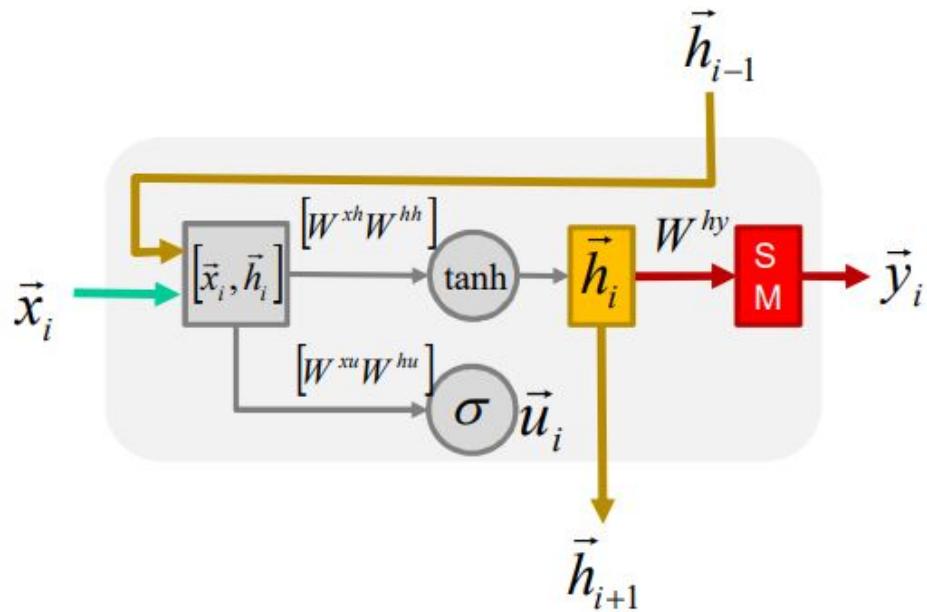
$$\hat{y}_i = W^{cy} \vec{h}_i$$

$$\vec{y}_i = SMAX(\hat{y}_i)$$

GRU (Gated Recurrent Unit)

Modif 2

$$\sigma = \text{sigmoid}$$



$$\vec{u}_i = \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1})$$

$$\vec{h}_i = \tanh(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1})$$

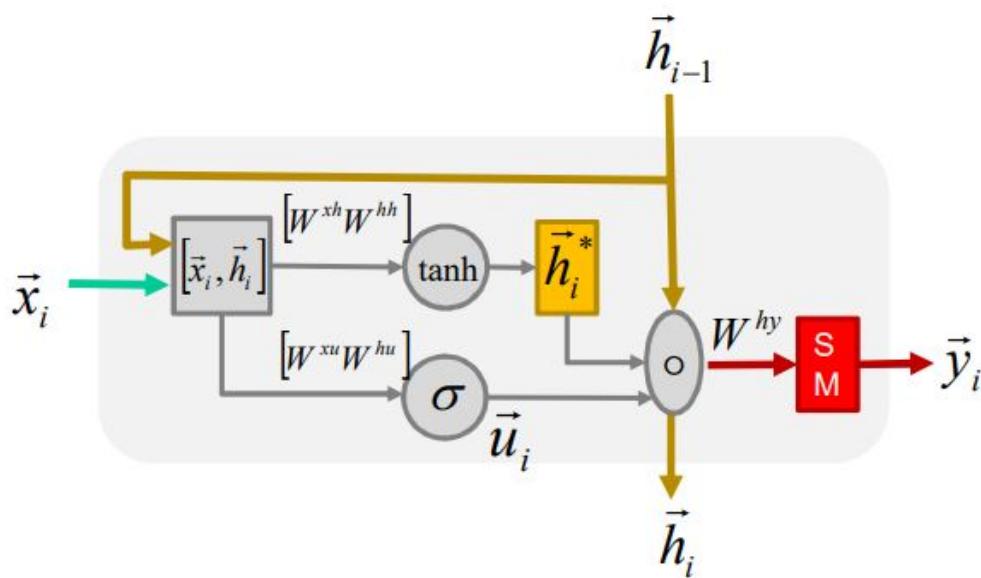
$$\hat{y}_i = W^{cy} \vec{h}_i$$

$$\bar{y}_i = \text{SMAX}(\hat{y}_i)$$

GRU (Gated Recurrent Unit)

Modif 2

Update gate



$$\vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1})$$

$$\vec{h}^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

$$\vec{h}_i = \vec{u}_i \circ \vec{h}^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1}$$

$$\hat{y}_i = W^{cy}\vec{h}_i$$

$$\bar{y}_i = SMAX(\hat{y}_i)$$

GRU (Gated Recurrent Unit)

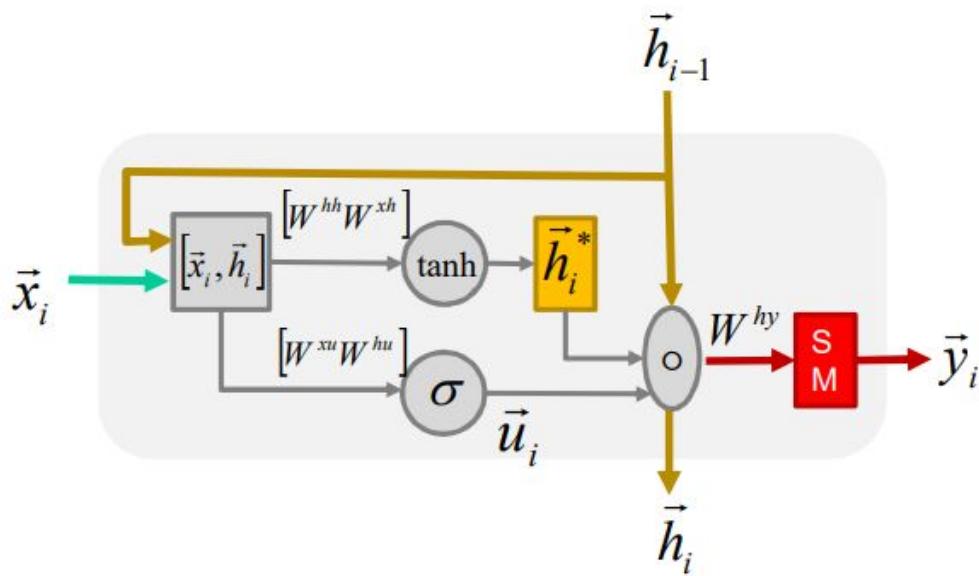
Modif 2

« element-wise product »

ou

« Hadamard product »

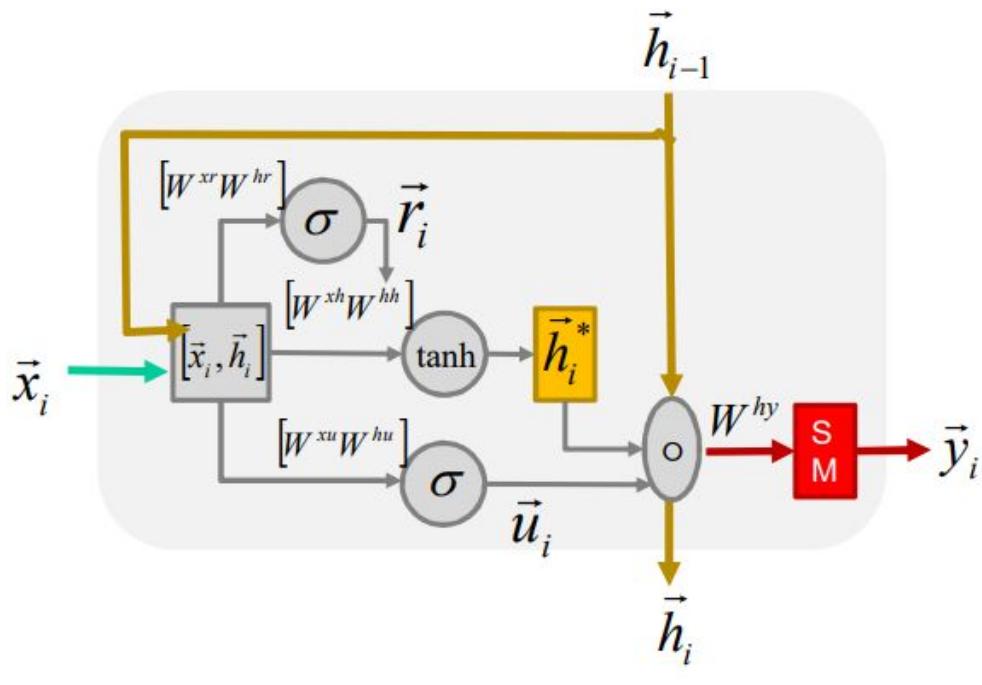
$$(a, b, c) \circ (x, y, z) = (ax, by, cz)$$



GRU (Gated Recurrent Unit)

Modif 3

Reset gate



$$\vec{u}_i = \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1})$$

$$\vec{r}_i = \sigma(W^{xr} \vec{x}_i + W^{hr} \vec{h}_{i-1})$$

$$\vec{h}^* = \tanh(W^{xh} \vec{x}_i + W^{hh} (\vec{r}_i \circ \vec{h}_{i-1}))$$

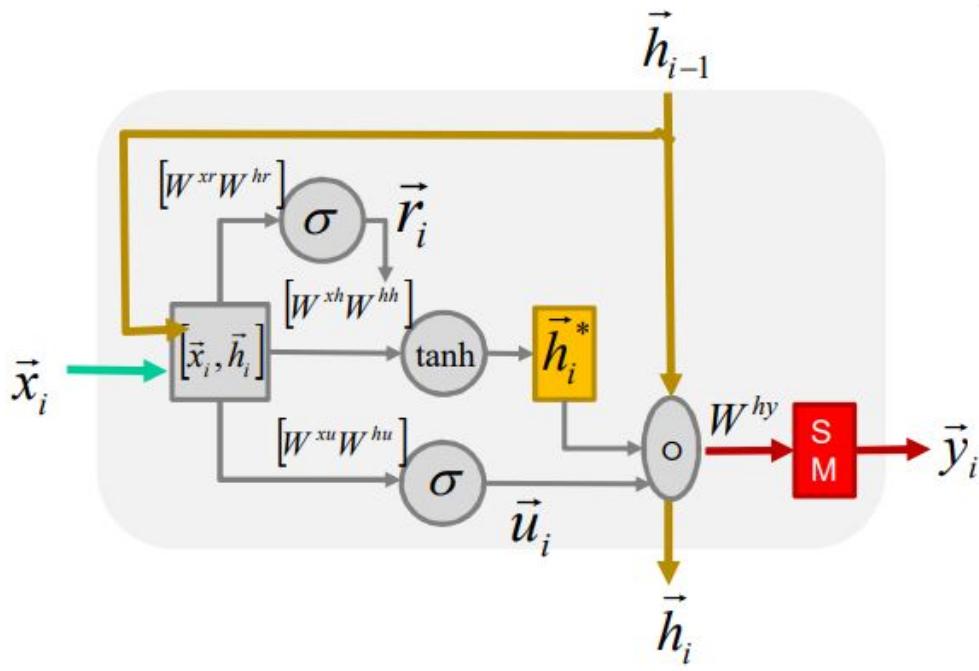
$$\vec{h}_i = \vec{u}_i \circ \vec{h}^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1}$$

$$\hat{y}_i = W^{cy} \vec{h}_i$$

$$\vec{y}_i = SMAX(\hat{y}_i)$$

Comprendre les gates

$$SI \begin{cases} u_i \\ r_i \end{cases} = 1 \quad \left(\quad \right)$$



$$\vec{u}_i = \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1})$$

$$\vec{r}_i = \sigma(W^{xr} \vec{x}_i + W^{hr} \vec{h}_{i-1})$$

$$\vec{h}^* = \tanh(W^{xh} \vec{x}_i + W^{hh} (\vec{r}_i \circ \vec{h}_{i-1}))$$

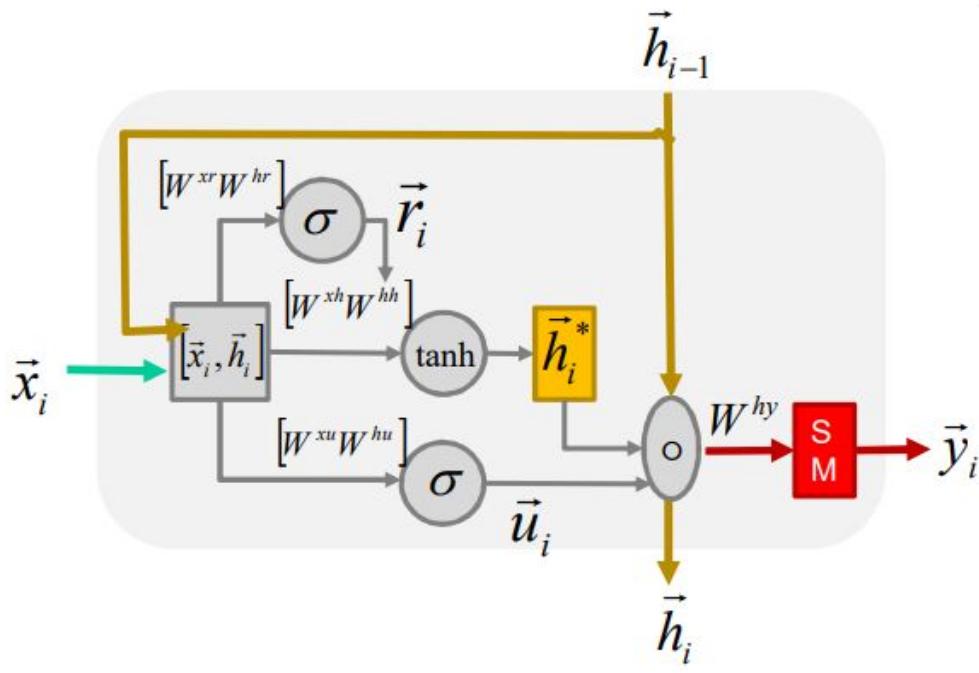
$$\vec{h}_i = \vec{u}_i \circ \vec{h}^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1}$$

$$\hat{y}_i = W^{cy} \vec{h}_i$$

$$\vec{y}_i = SMAX(\hat{y}_i)$$

Comprendre les gates

$$SI \begin{cases} u_i \\ r_i \end{cases} = 1 \quad \left(\quad \right)$$



$$\vec{u}_i = \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1})$$

$$\vec{r}_i = \sigma(W^{xr} \vec{x}_i + W^{hr} \vec{h}_{i-1})$$

$$\vec{h}^* = \tanh(W^{xh} \vec{x}_i + W^{hh} (\vec{r}_i \circ \vec{h}_{i-1}))$$

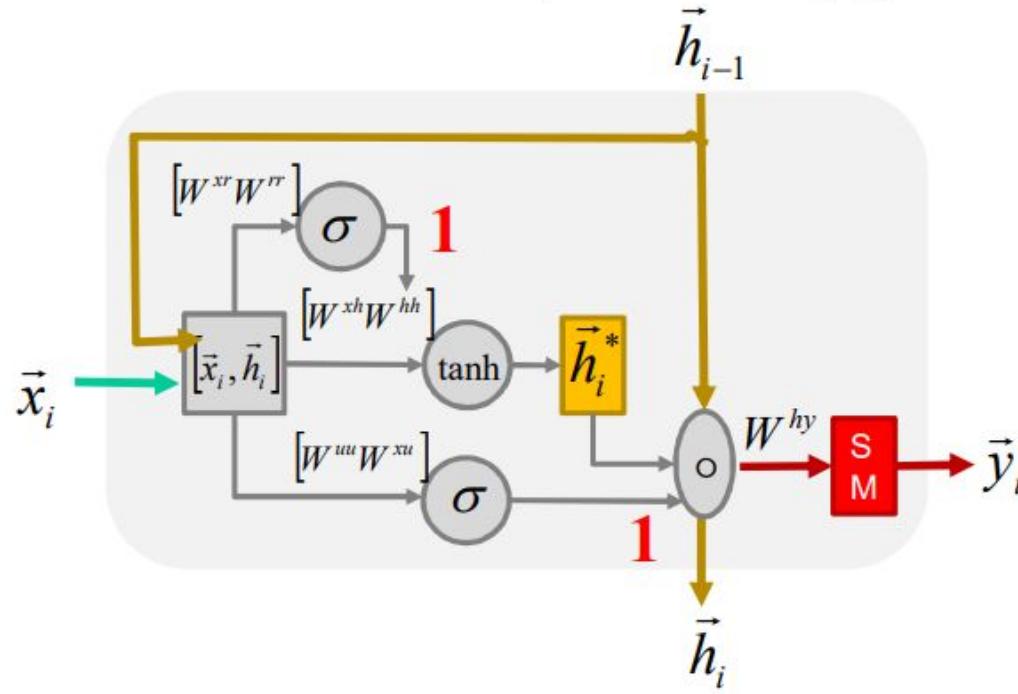
$$\vec{h}_i = \vec{u}_i \circ \vec{h}^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1}$$

$$\hat{y}_i = W^{cy} \vec{h}_i$$

$$\vec{y}_i = SMAX(\hat{y}_i)$$

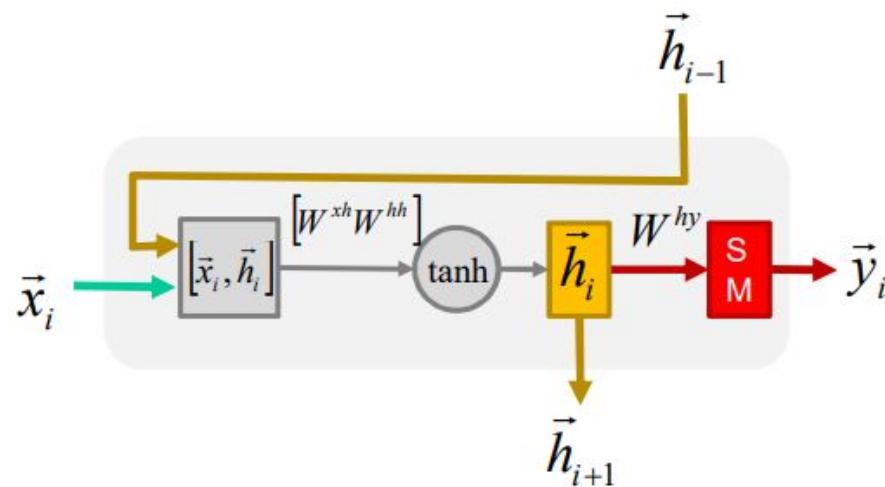
Comprendre les gates

$$SI \quad \begin{aligned} \vec{u}_i &= 1 \\ \vec{r}_i &= 1 \end{aligned} \quad \left\{ \begin{array}{l} \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \vec{h}_i = \vec{h}_i^* \\ \hat{y}_i = W^{cy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i) \end{array} \right.$$



Comprendre les gates

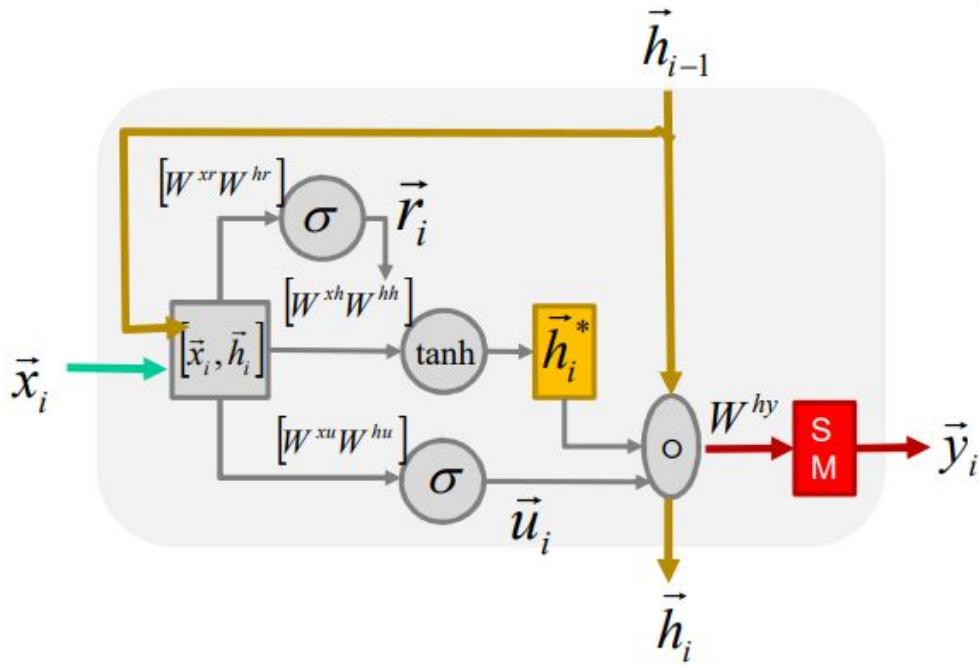
$$SI \quad \begin{cases} \vec{u}_i = 1 \\ \vec{r}_i = 1 \end{cases} \quad \left\{ \begin{array}{l} \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \vec{h}_i = \vec{h}_i^* \\ \hat{y}_i = W^{cy}\vec{h}_i \\ \bar{y}_i = SMAX(\hat{y}_i) \end{array} \right.$$



RNN de base!

Comprendre les gates

$$SI \begin{cases} u_i & \neq 1 \\ r_i & = 0 \end{cases}$$



$$\vec{u}_i = \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1})$$

$$\vec{r}_i = \sigma(W^{xr} \vec{x}_i + W^{hr} \vec{h}_{i-1})$$

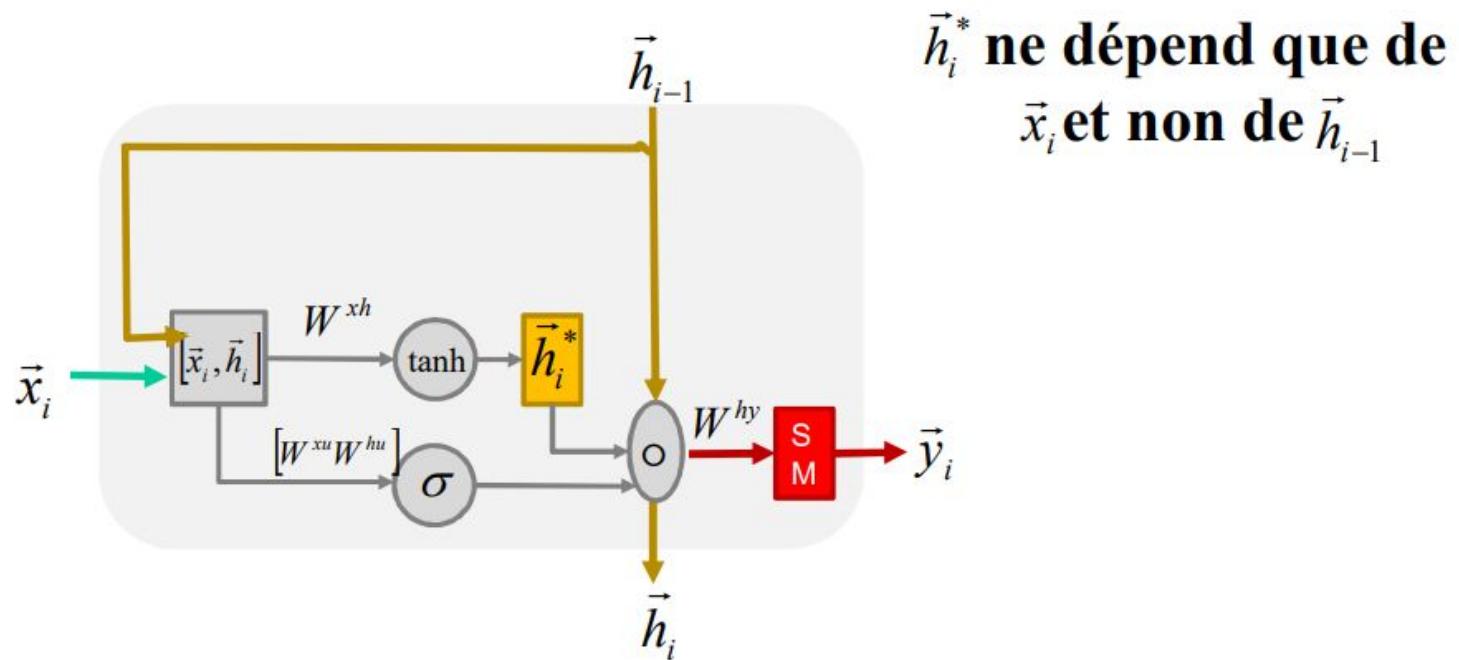
$$\vec{h}_i^* = \tanh(W^{xh} \vec{x}_i + W^{hh} (\vec{r}_i \circ \vec{h}_{i-1}))$$

$$\vec{h}_i = \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1}$$

$$\hat{y}_i = W^{cy} \vec{h}_i$$

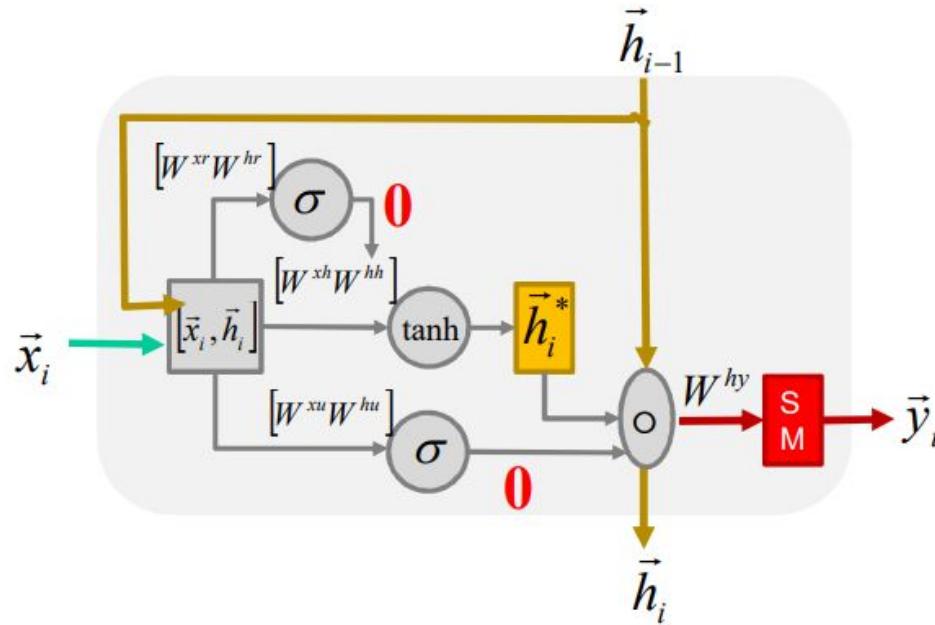
$$\vec{y}_i = SMAX(\hat{y}_i)$$

$$SI \quad \begin{cases} \vec{u}_i \neq 1 \\ \vec{r}_i = 0 \end{cases} \quad \left\{ \begin{array}{l} \vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\ \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i) \\ \vec{h}_i = \vec{u}_i \otimes \vec{h}_i^* + (1 - \vec{u}_i) \otimes \vec{h}_{i-1} \\ \hat{y}_i = W^{cy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i) \end{array} \right.$$



$$SI \quad \begin{cases} \vec{u}_i = 0 \\ \vec{r}_i = 0 \end{cases}$$

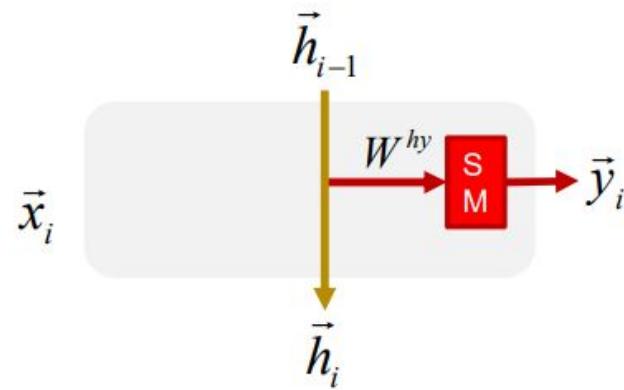
$$\begin{aligned}\vec{u}_i &= \sigma(W^{ru} \vec{x}_i + W^{rh} \vec{h}_{i-1}) \\ \vec{r}_i &= \sigma(W^{rr} \vec{x}_i + W^{rh} \vec{h}_{i-1}) \\ \vec{h}^*_i &= \tanh(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1}) \\ \vec{h}_i &= \vec{u}_i \odot \vec{h}^*_i + (1 - \vec{u}_i) \odot \vec{h}_{i-1} \\ \hat{y}_i &= W^{cy} \vec{h}_i \\ \bar{y}_i &= SMAX(\hat{y}_i)\end{aligned}$$



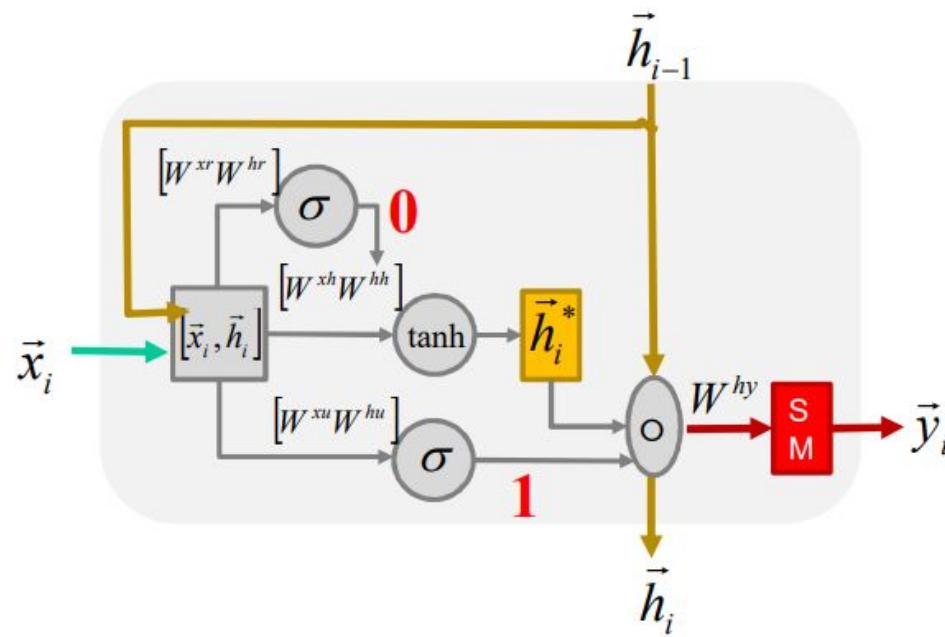
$$SI \quad \begin{cases} \vec{u}_i = 0 \\ \vec{r}_i = 0 \end{cases} \quad \left\{ \begin{array}{l} \vec{h}_i = \vec{h}_{i-1} \\ \hat{y}_i = W^{cy} \vec{h}_i \\ \bar{y}_i = SMAX(\hat{y}_i) \end{array} \right.$$

\vec{h}_{i-1} est recopié
 \vec{x}_i est ignoré

Aucune disparition de gradient

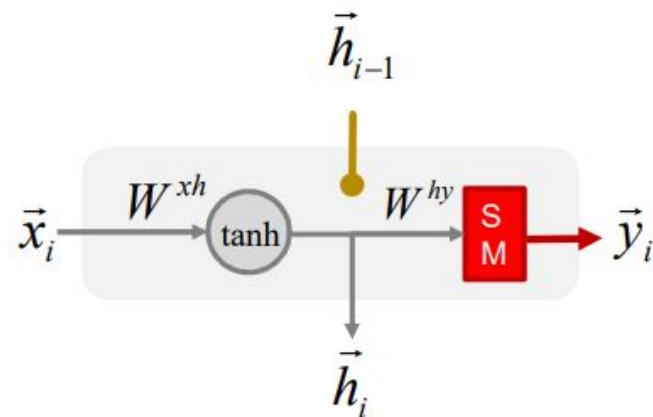


$$\left. \begin{array}{l} SI \\ \vec{u}_i = 1 \\ \vec{r}_i = 0 \end{array} \right\} \quad \begin{aligned}
 \vec{u}_i &= \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\
 \vec{r}_i &= \sigma(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}) \\
 \vec{h}_i^* &= \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\
 \vec{h}_i &= \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{r}_i) \circ \vec{h}_{i-1} \\
 \hat{y}_i &= W^{cy}\vec{h}_i \\
 \vec{y}_i &= SMAX(\hat{y}_i)
 \end{aligned}$$

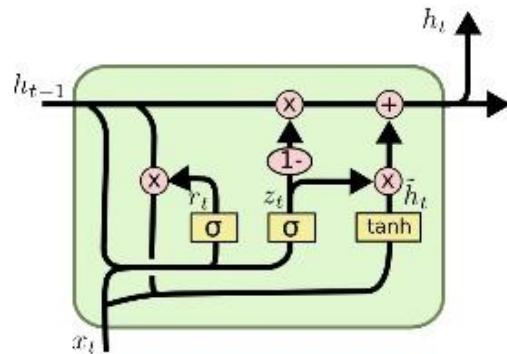


$$SI \quad \begin{cases} \vec{u}_i = 1 \\ \vec{r}_i = 0 \end{cases} \quad \left\{ \begin{array}{l} \vec{h}_i = \tanh(W^{xh}\vec{x}_i) \\ \hat{y}_i = W^{cy}\vec{h}_i \\ \bar{y}_i = SMAX(\hat{y}_i) \end{array} \right.$$

\vec{h}_{i-1} est ignoré
 \vec{x}_i est le seul utilisé
Gradient temporel bloqué



Gated Recurrent Unit (GRU)



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

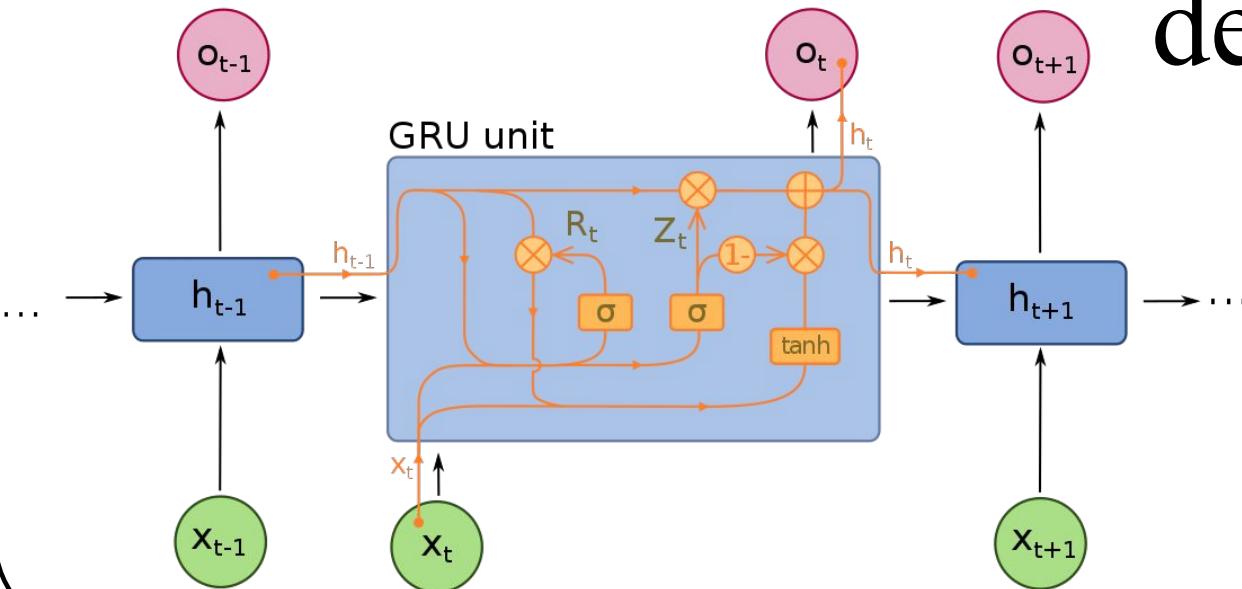
$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

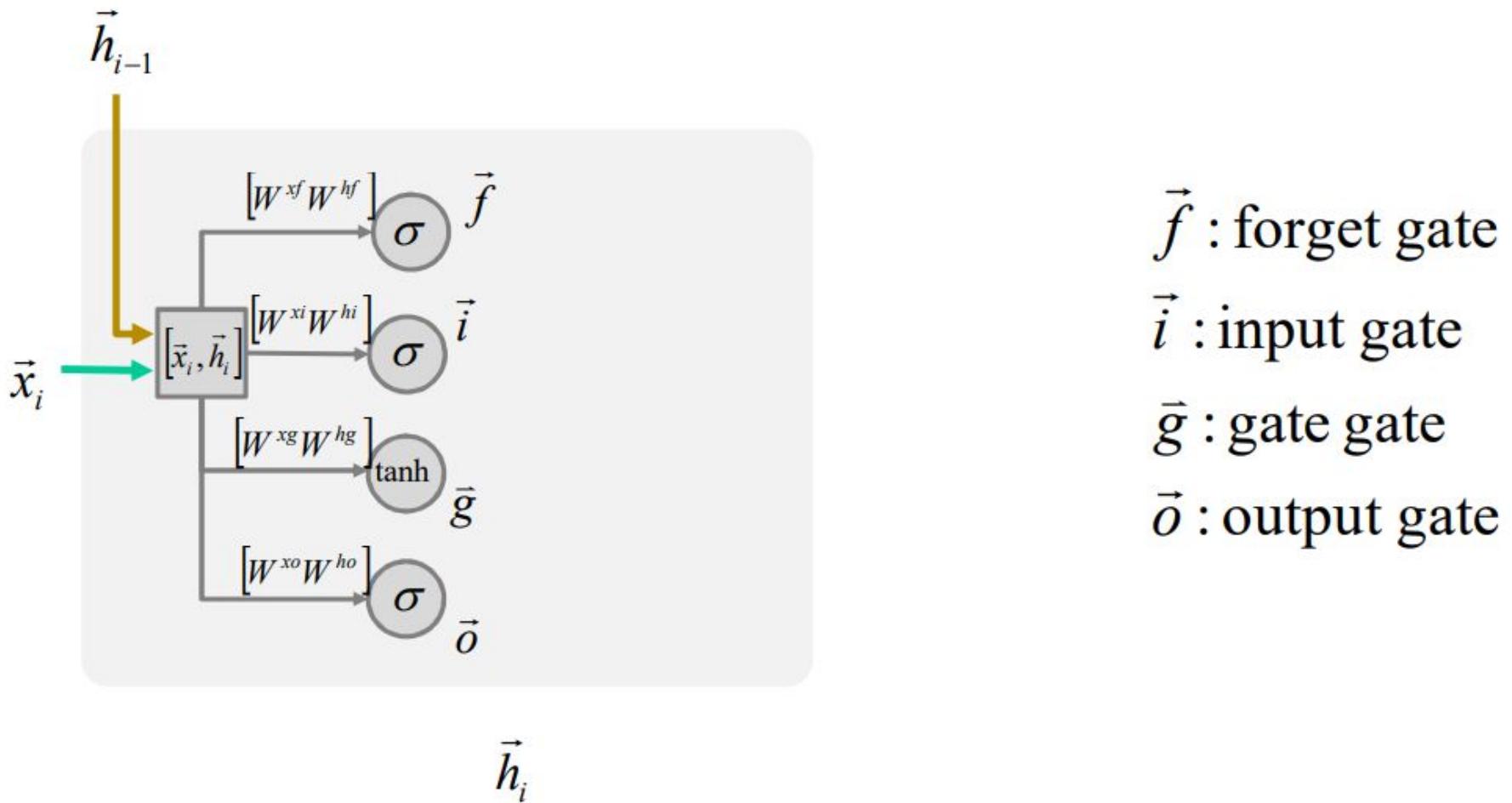
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Le web regorge
d'illustrations
de GRU!

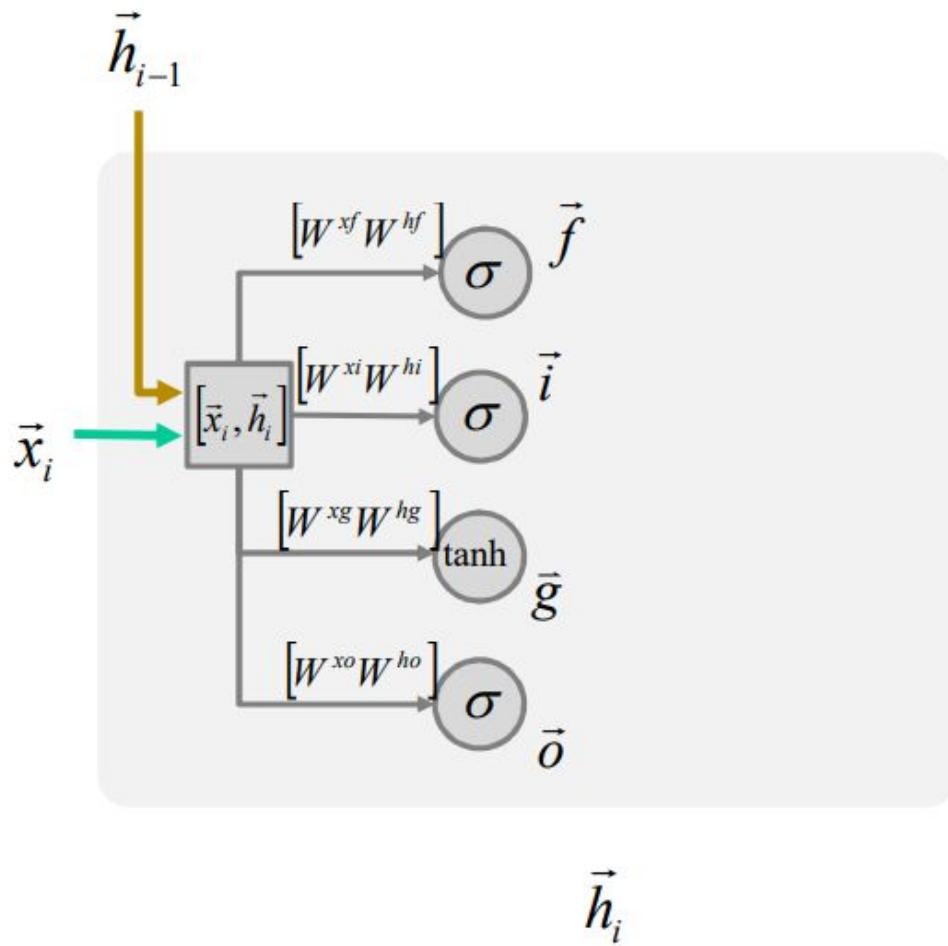
Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes



LSTM (Long Short Term Memory)



LSTM (Long Short Term Memory)



\vec{f} : forget gate

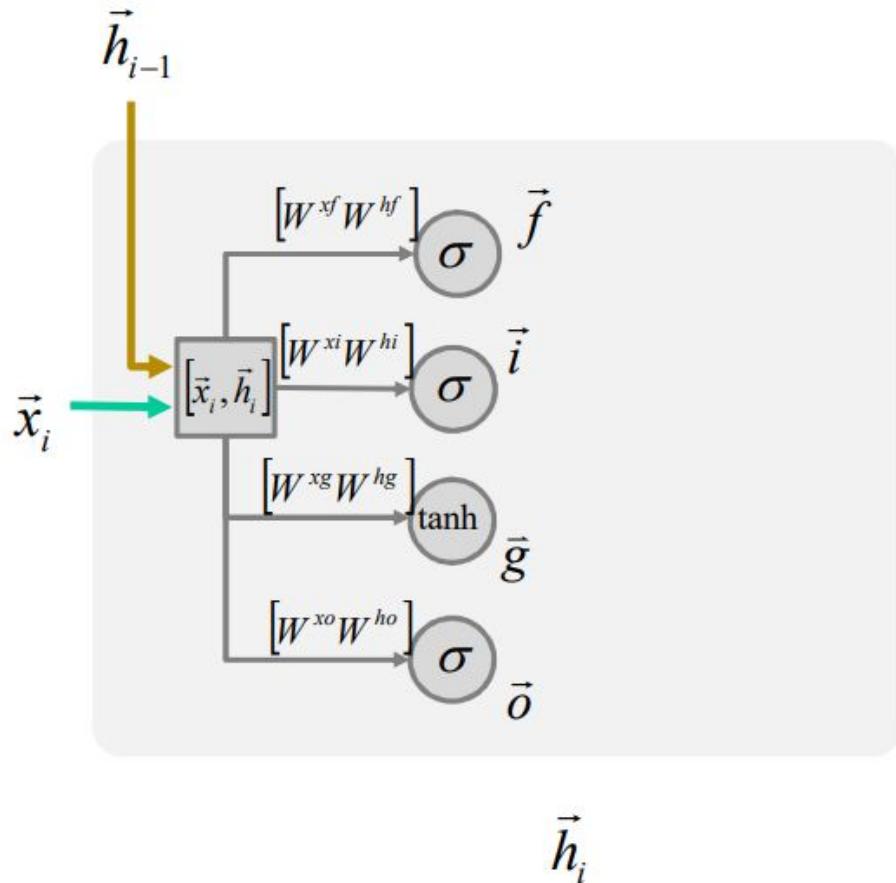
\vec{i} : input gate

\vec{g} : gate candidate

\vec{o} : output gate

Modèle récurrent
le plus utilisé.
À bien
comprendre !

LSTM (Long Short Term Memory)



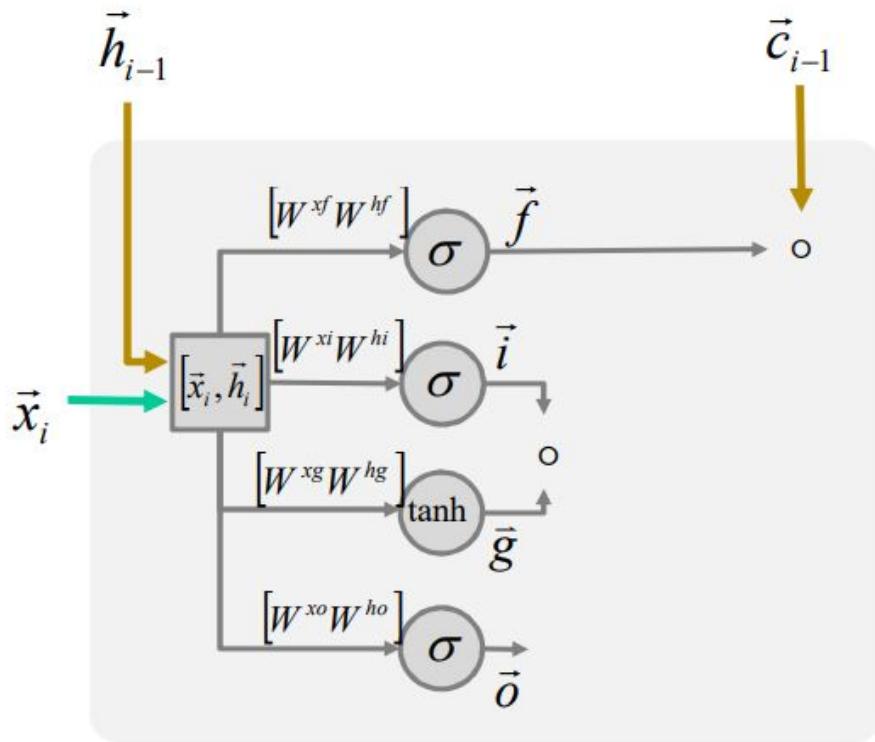
$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

$$\bar{g} = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

$$\vec{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

LSTM (Long Short Term Memory)



$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

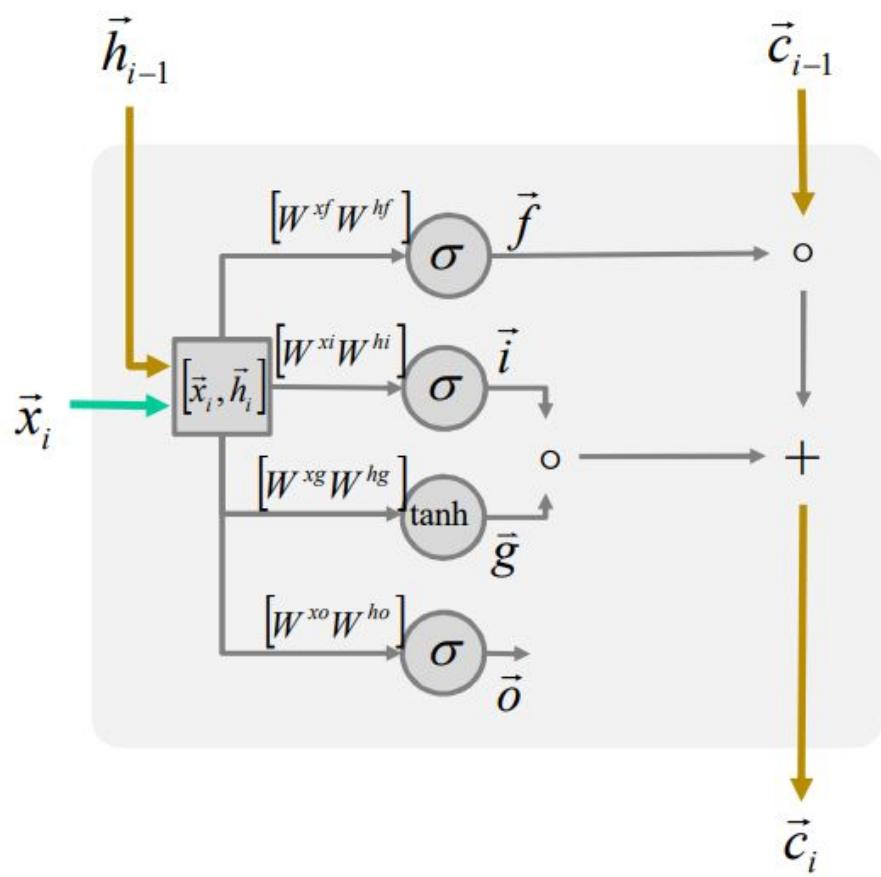
$$\vec{g} = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

$$\vec{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

$$\vec{f} \circ \vec{c}_i$$

$$\vec{i} \circ \vec{g}$$

LSTM (Long Short Term Memory)



$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

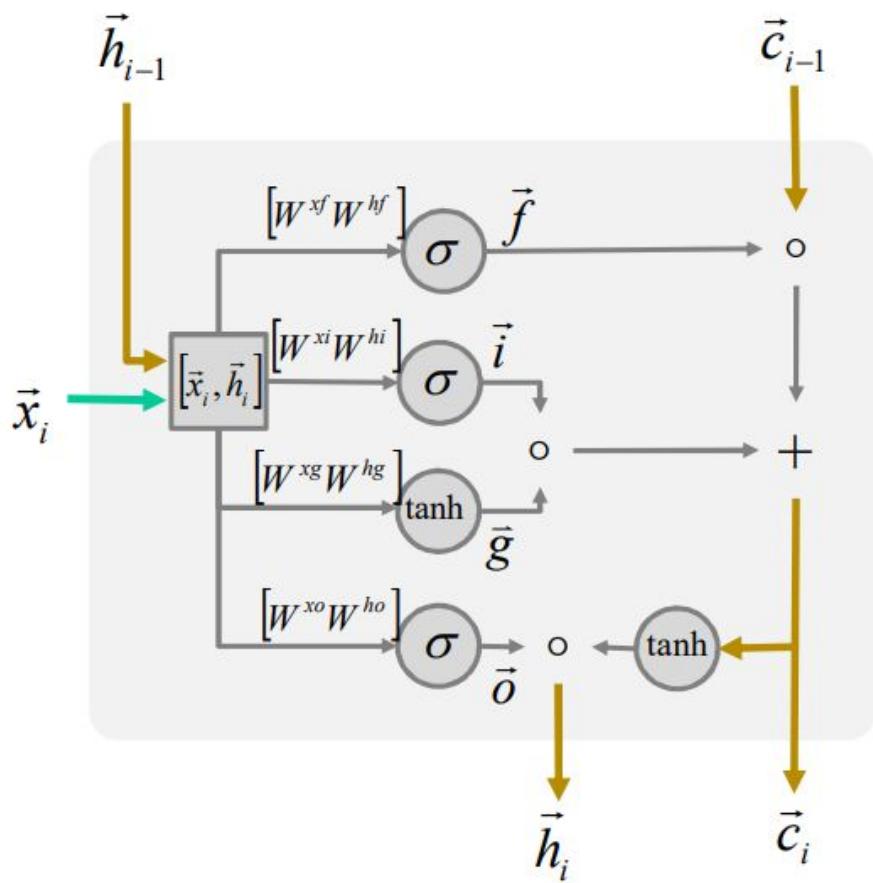
$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

$$\vec{g} = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

$$\vec{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

$$\vec{c}_i = \vec{f} \circ \vec{c}_{i-1} + \vec{i} \circ \vec{g}$$

LSTM (Long Short Term Memory)



$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

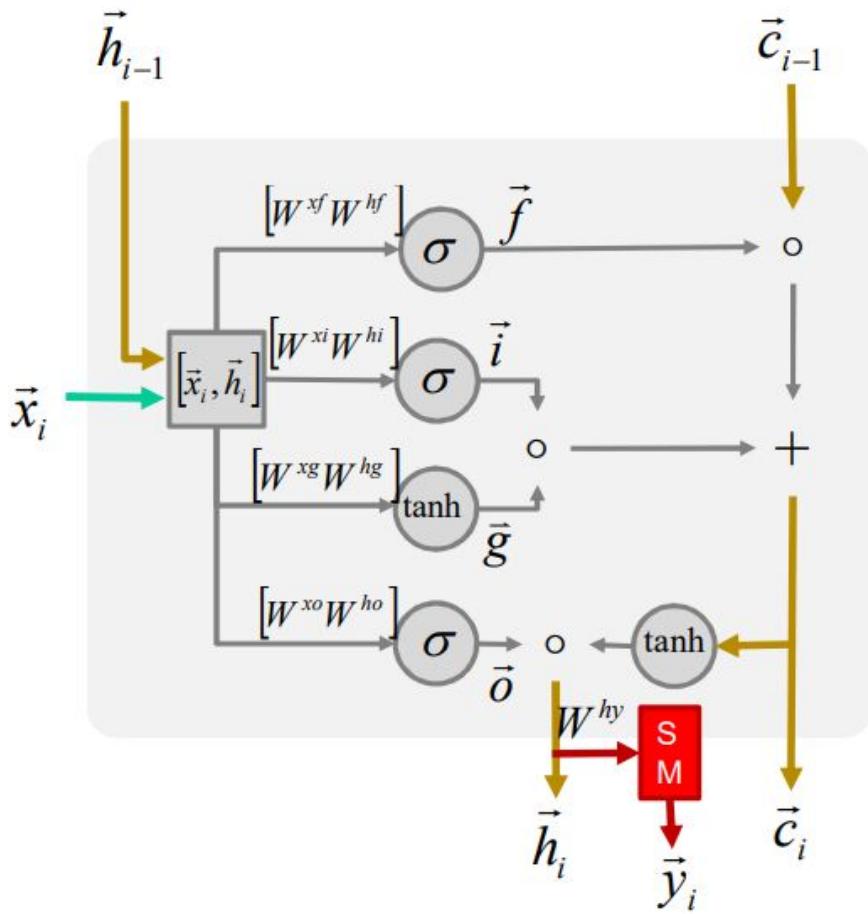
$$\vec{g} = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

$$\vec{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

$$\vec{c}_i = \vec{f} \circ \vec{c}_{i-1} + \vec{i} \circ \vec{g}$$

$$\vec{h}_i = \vec{o} \circ \tanh(\vec{c}_i)$$

LSTM (Long Short Term Memory)



$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

$$\vec{g} = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

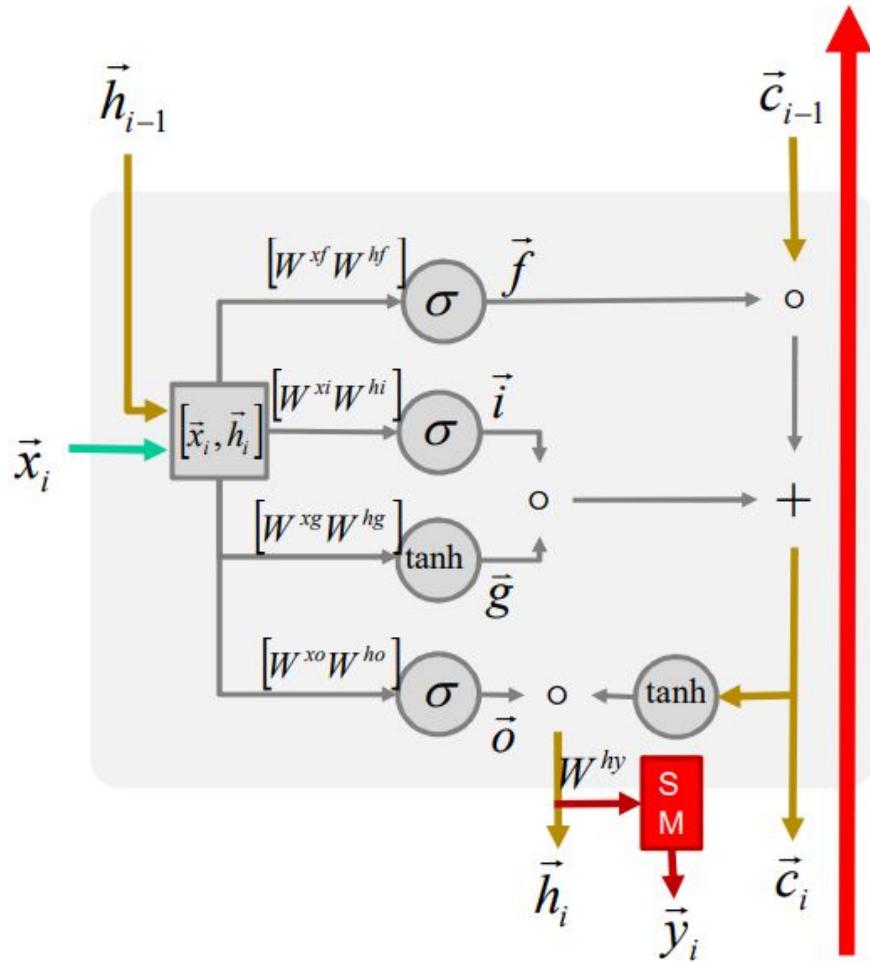
$$\vec{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

$$\vec{c}_i = \vec{f} \circ \vec{c}_{i-1} + \vec{i} \circ \vec{g}$$

$$\vec{h}_i = \vec{o} \circ \tanh(\vec{c}_i)$$

$$\vec{y}_i = \text{SMAX}(W^{hy} \vec{h}_i)$$

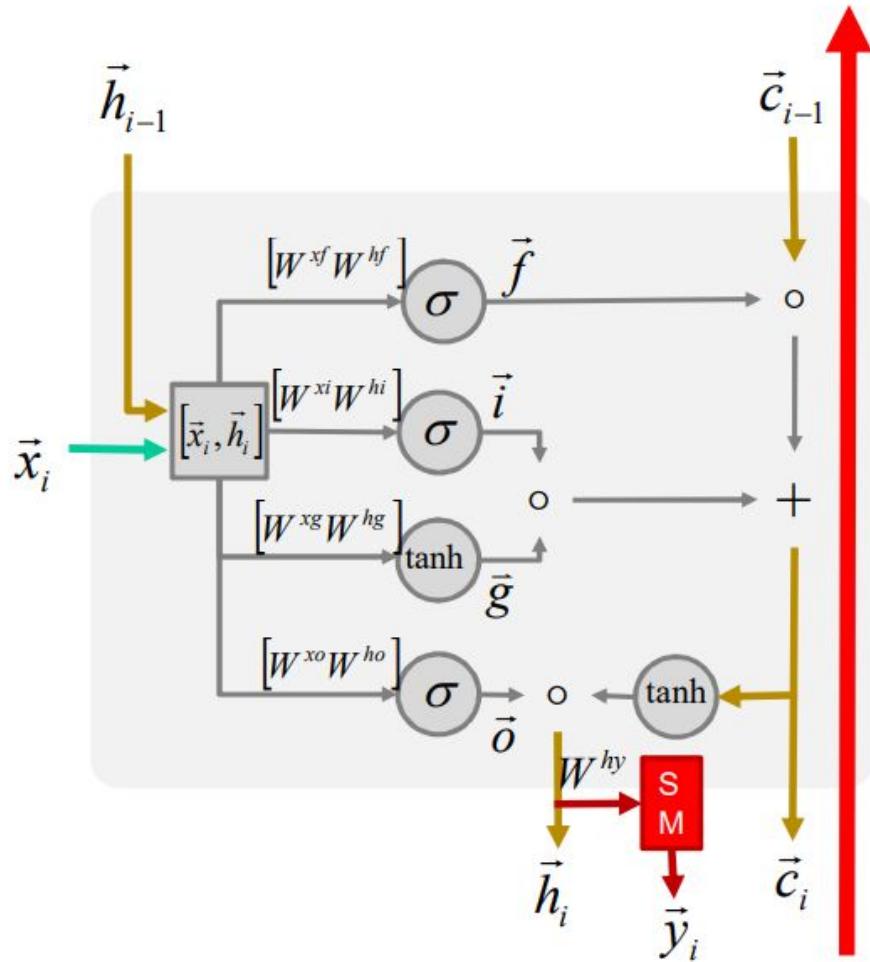
LSTM (Long Short Term Memory)



$$\begin{aligned}\vec{f} &= \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1}) \\ \vec{i} &= \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1}) \\ \vec{g} &= \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1}) \\ \vec{o} &= \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})\end{aligned}$$

Gradient simplifié

LSTM (Long Short Term Memory)



$$\vec{f} = \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1})$$

$$\vec{i} = \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1})$$

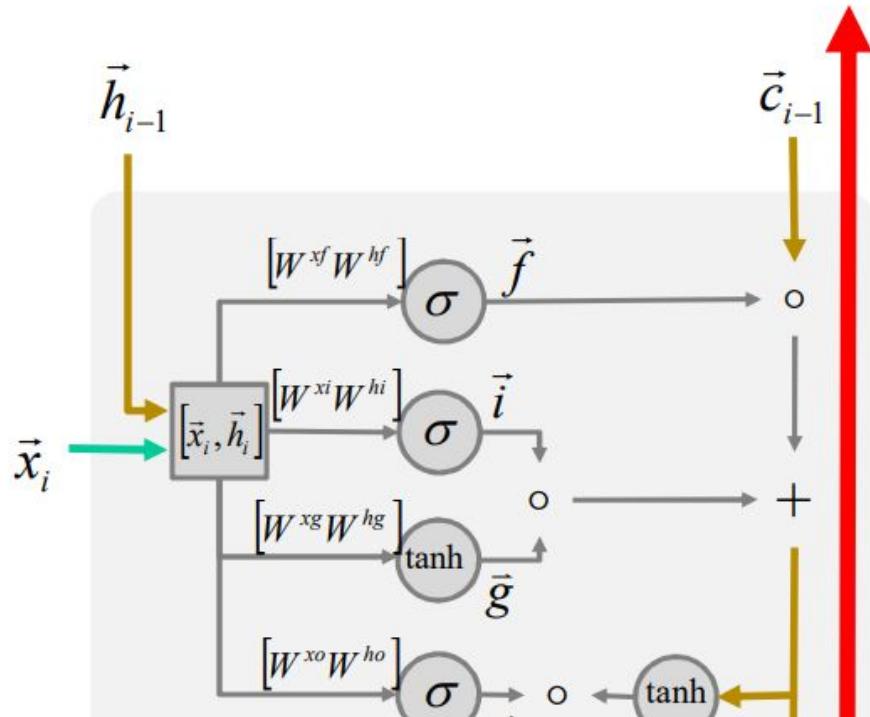
$$\bar{c}_i = \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1})$$

$$\bar{o} = \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})$$

Gradient simplifié

Ca vous rappelle quelque chose ?

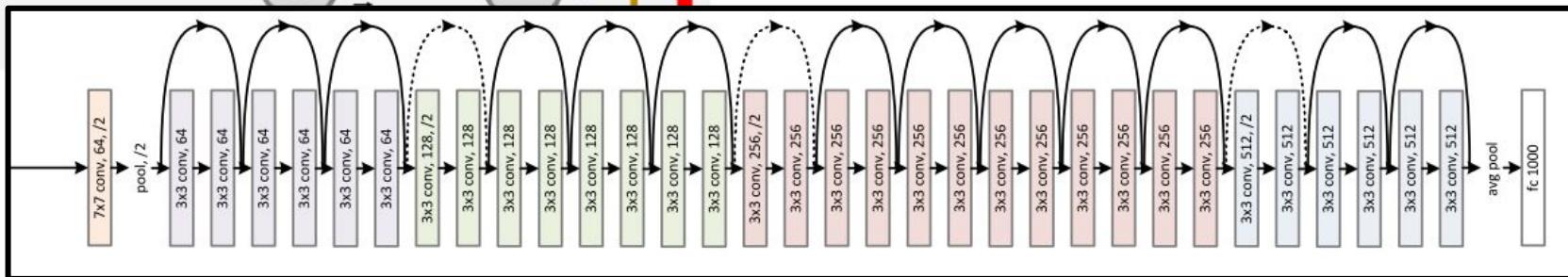
LSTM (Long Short Term Memory)



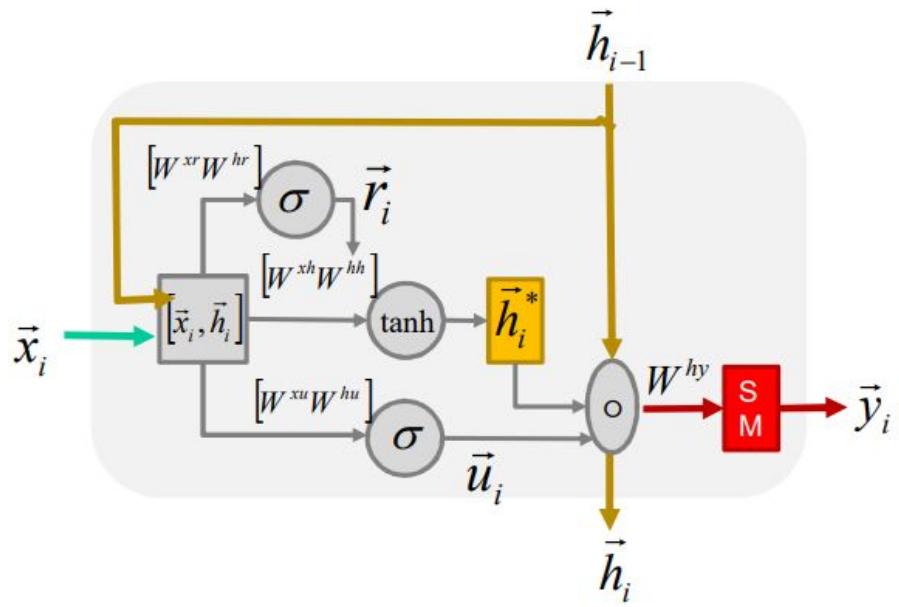
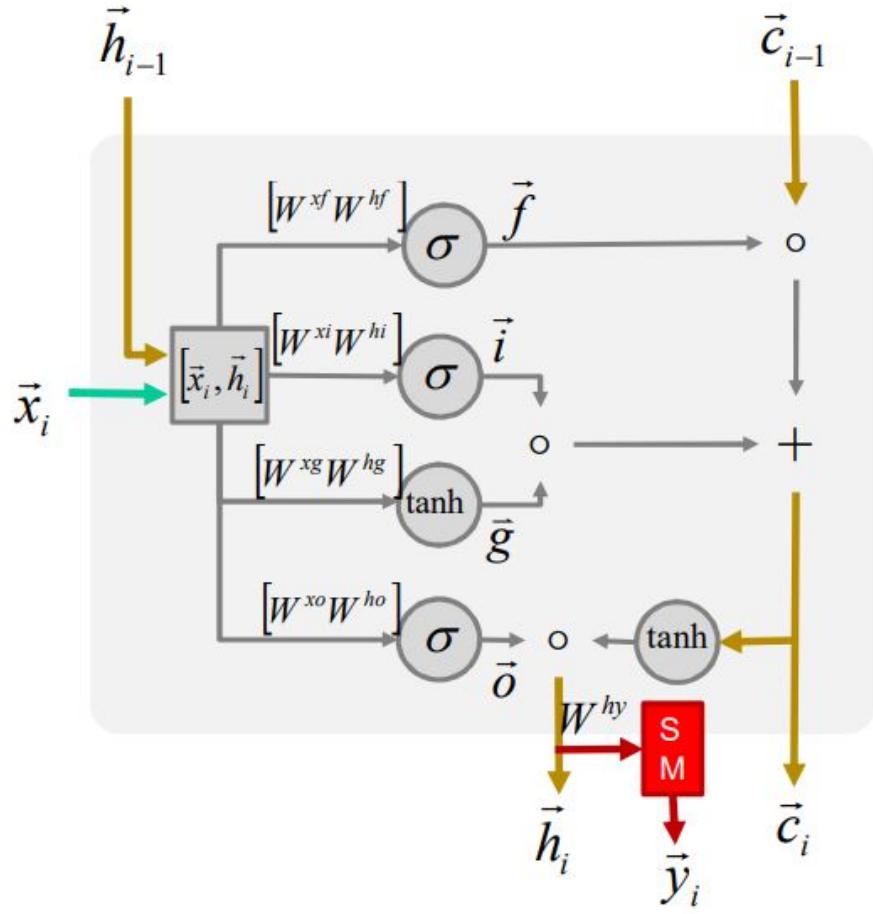
$$\begin{aligned}\vec{f} &= \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1}) \\ \vec{i} &= \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1}) \\ \bar{g} &= \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1}) \\ \bar{o} &= \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})\end{aligned}$$

Gradient simplifié

Ça vous rappelle quelque chose ?
Le Resnet !

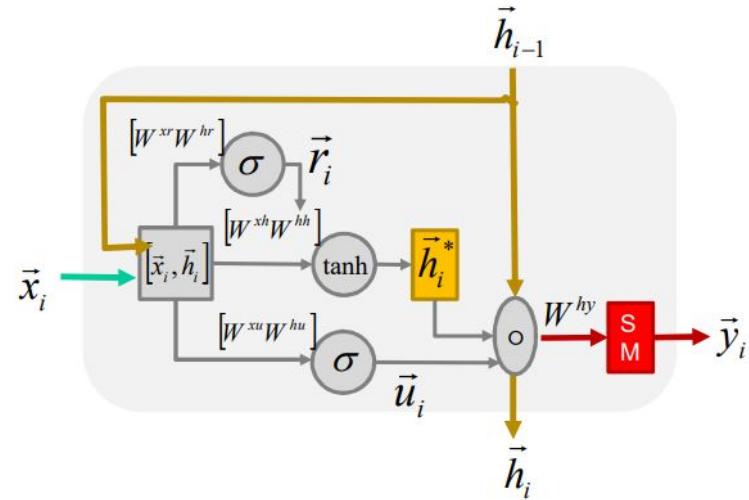
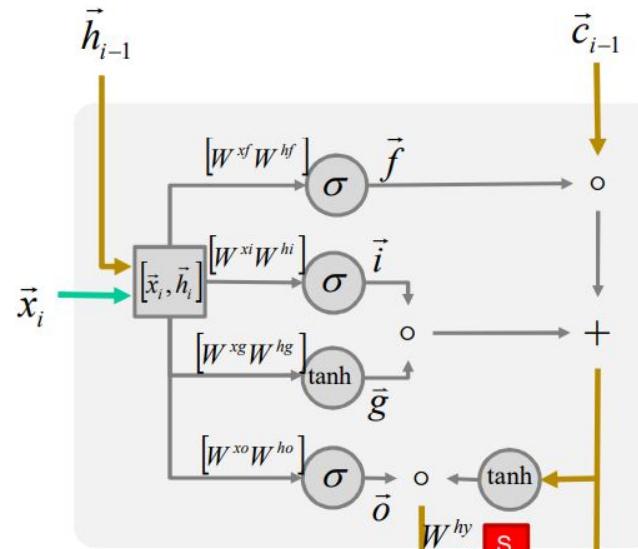


LSTM et GRU

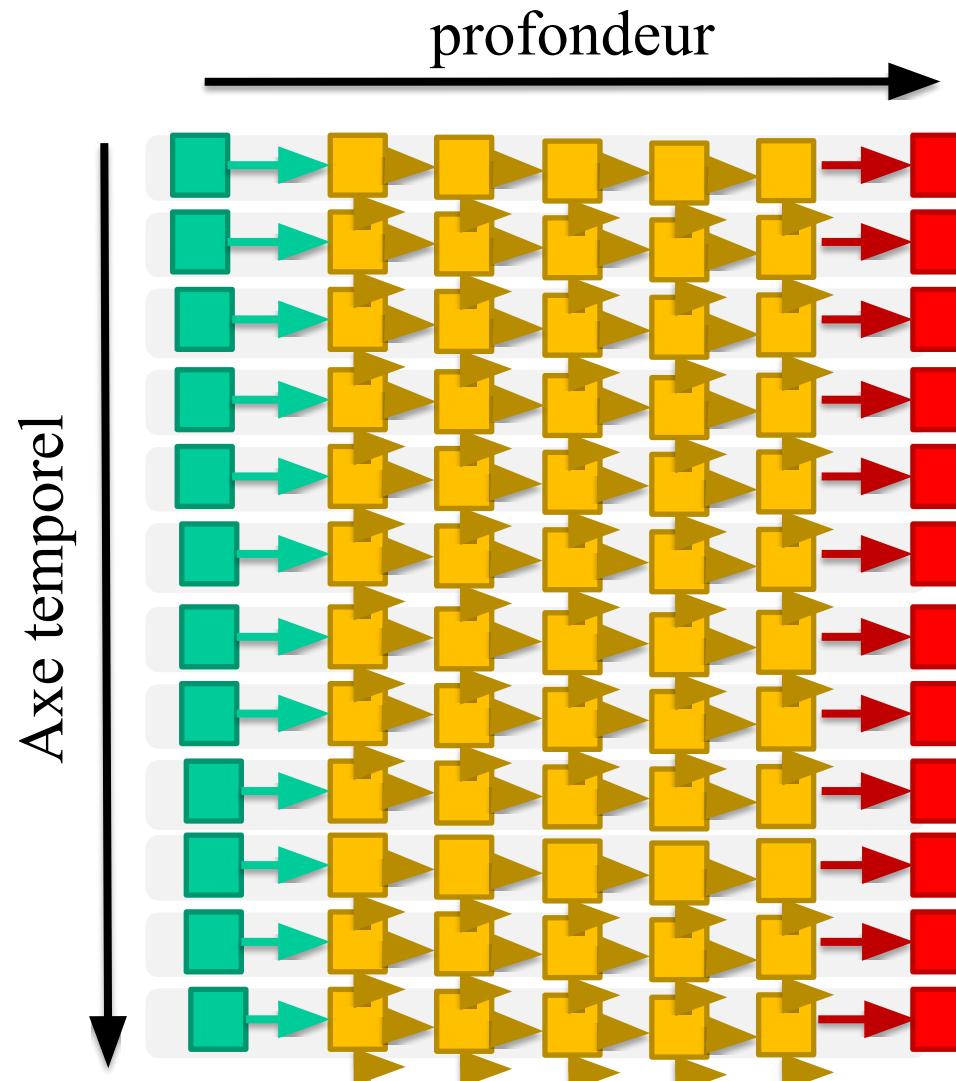


LSTM et GRU

- Servent à protéger le gradient
- Conçus empiriquement
- GRU légèrement plus simple
- Les “gates” ne servent qu’à bloquer ou permettre à l’information (données ou temporelle) de passer



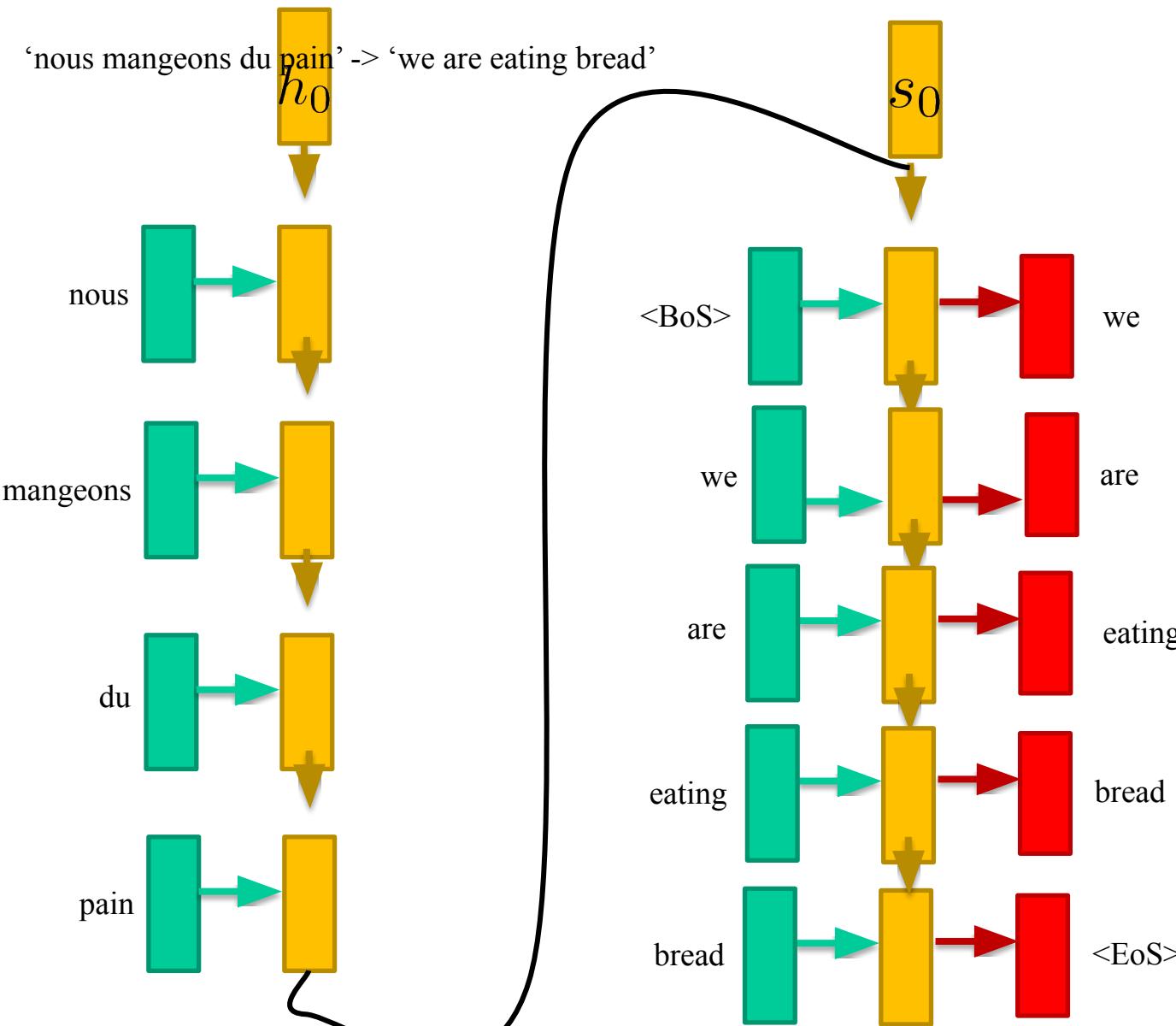
RNN multi-couches



Modèles d'attention

Seq2Seq:

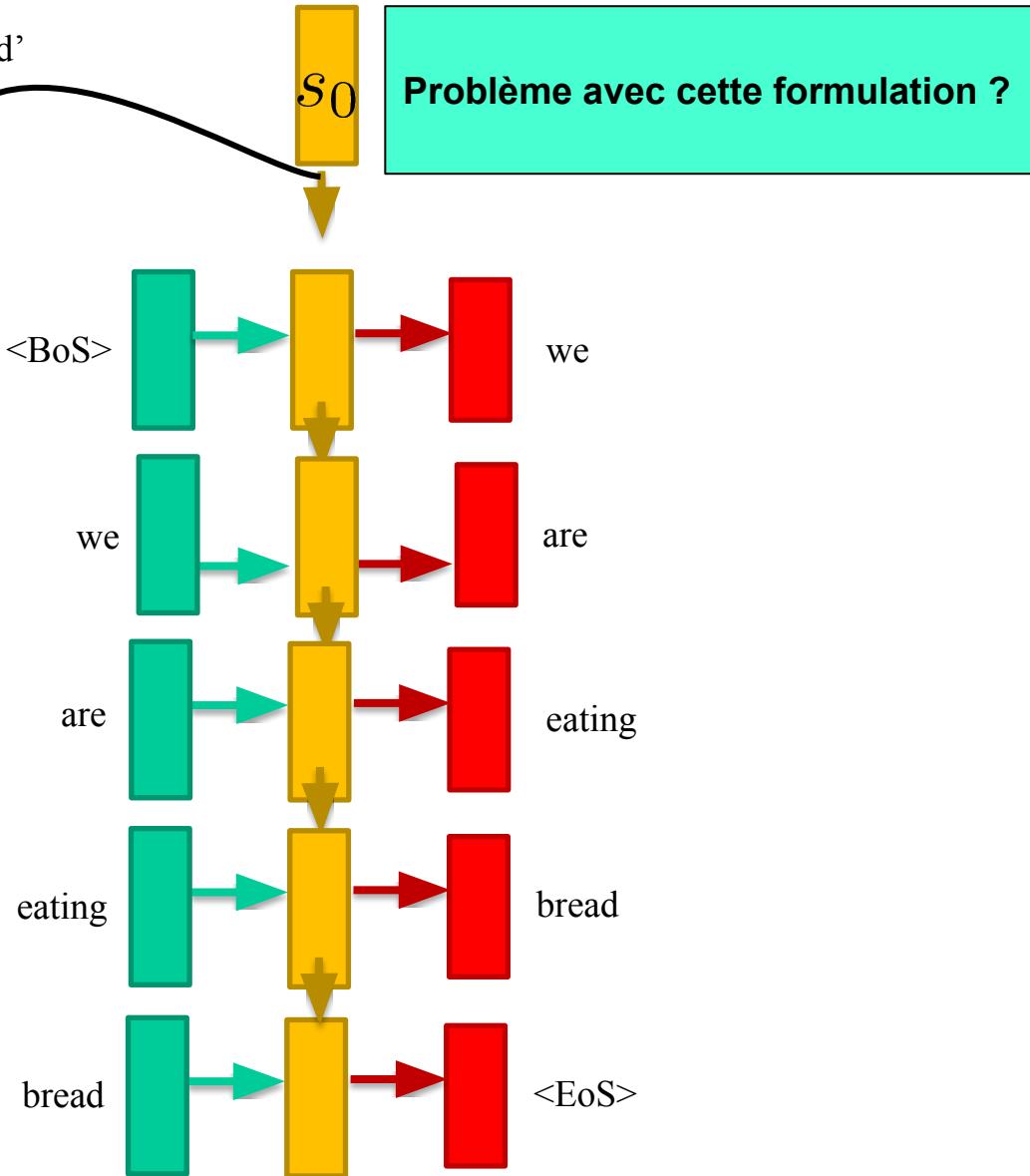
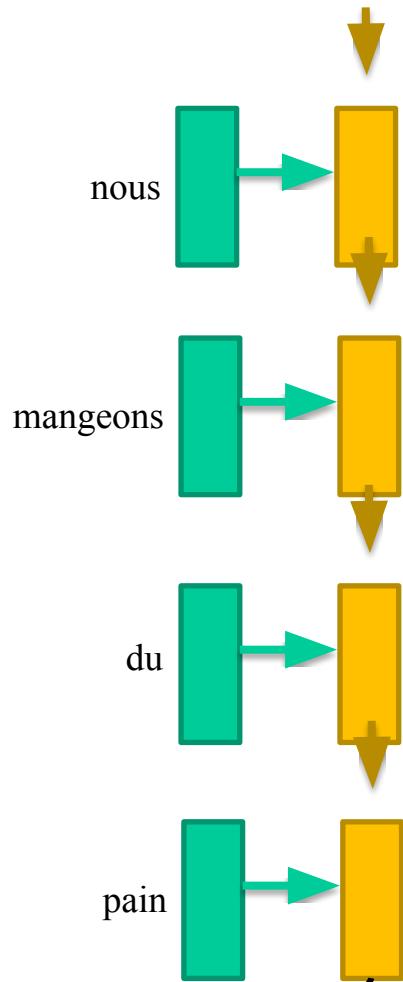
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.



Seq2Seq:

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

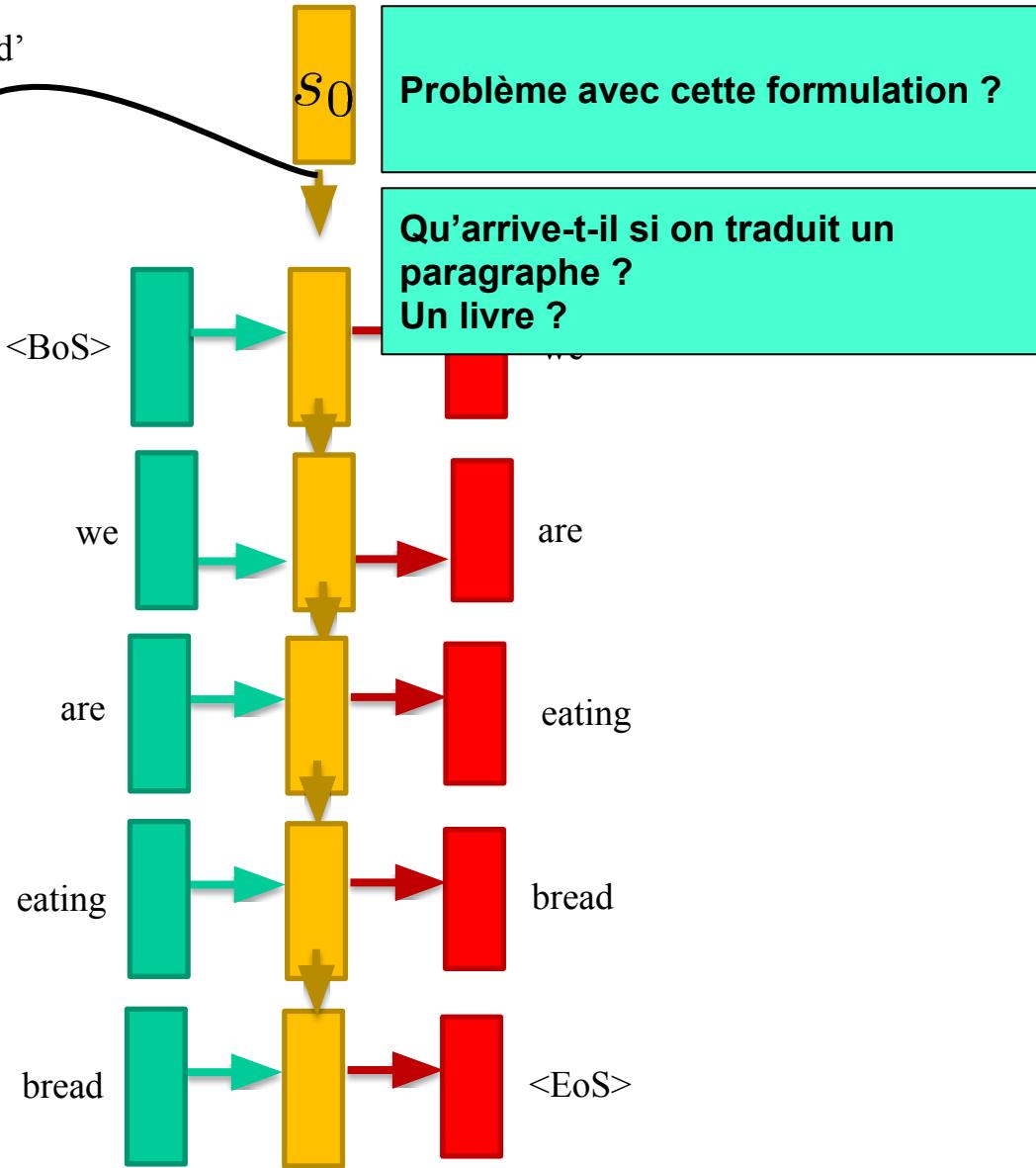
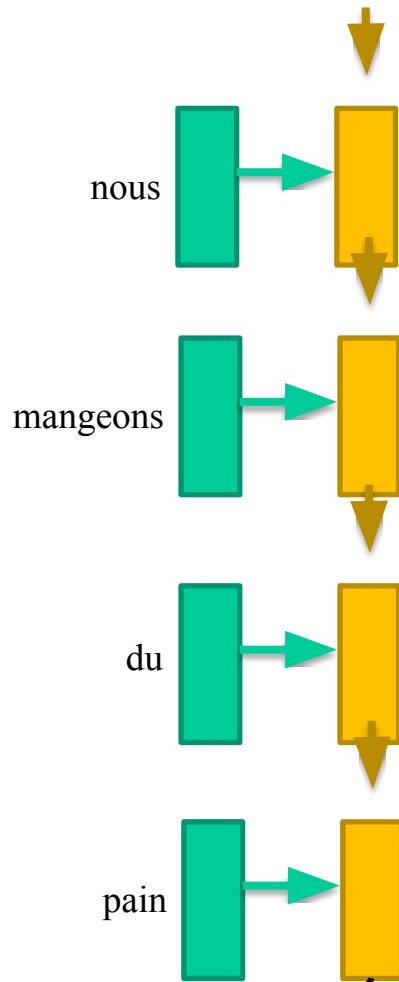
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq:

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

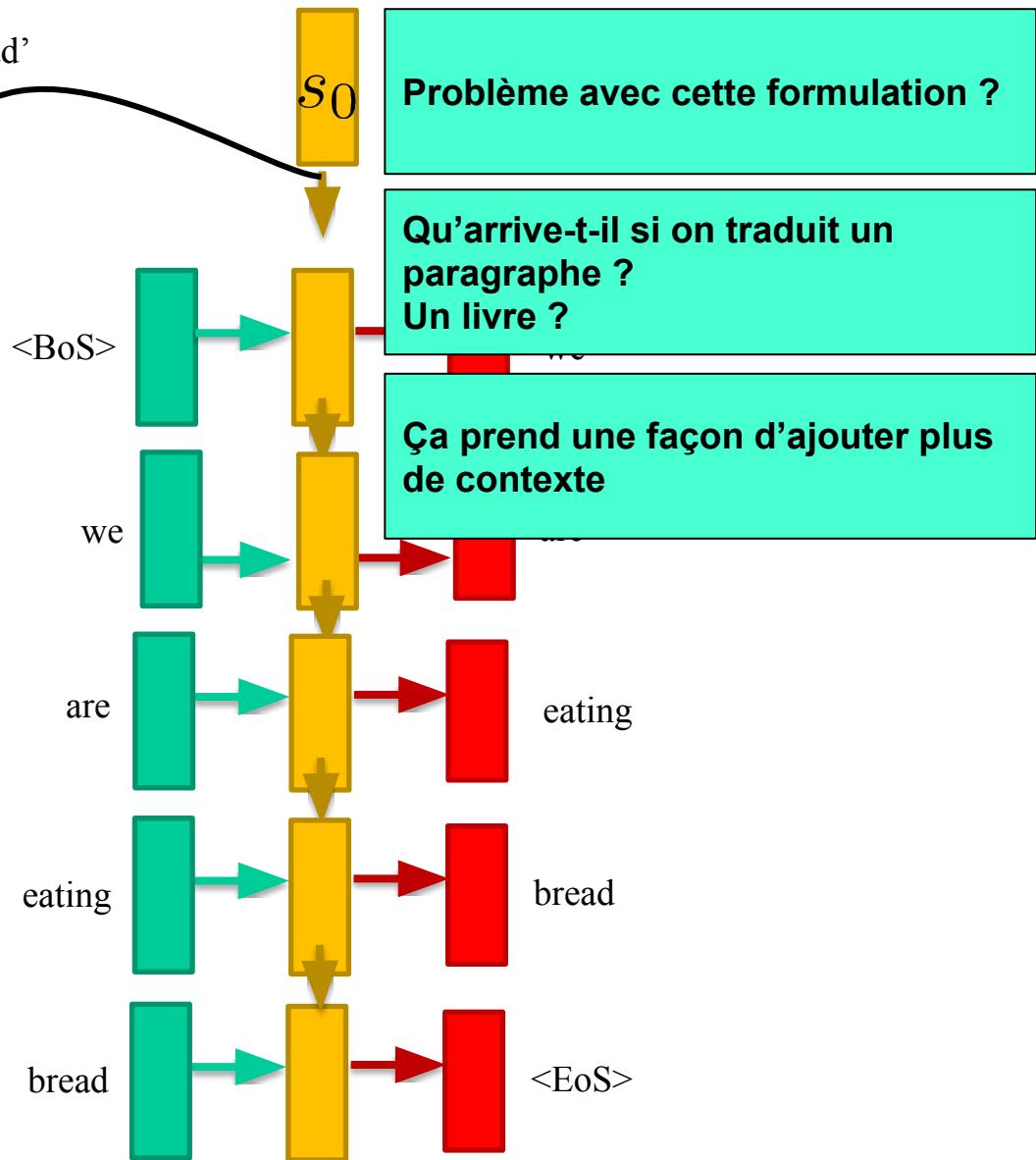
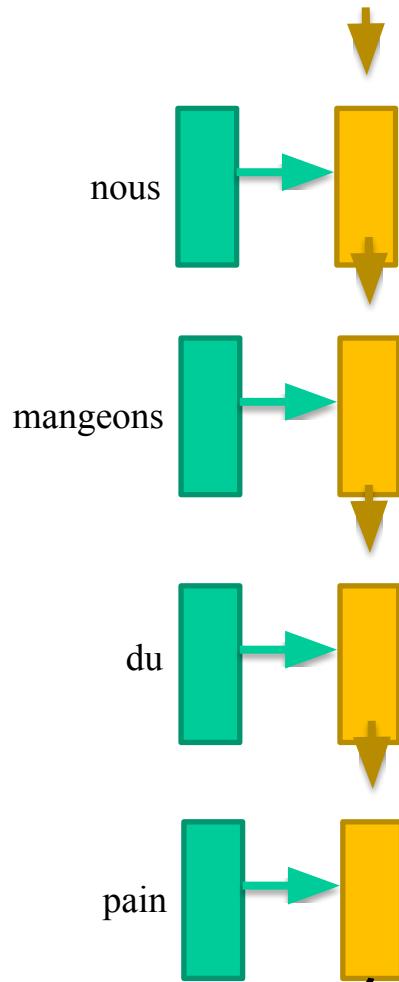
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq:

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

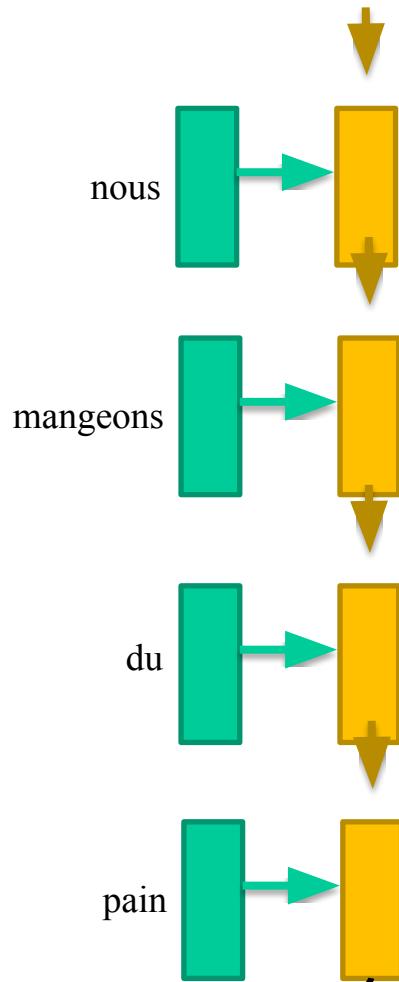
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq:

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

‘nous mangeons du pain’ -> ‘we are eating bread’



s_0

Problème avec cette formulation ?

Qu’arrive-t-il si on traduit un paragraphe ?
Un livre ?

Ça prend une façon d’ajouter plus de contexte

<BoS>

we

are

eating

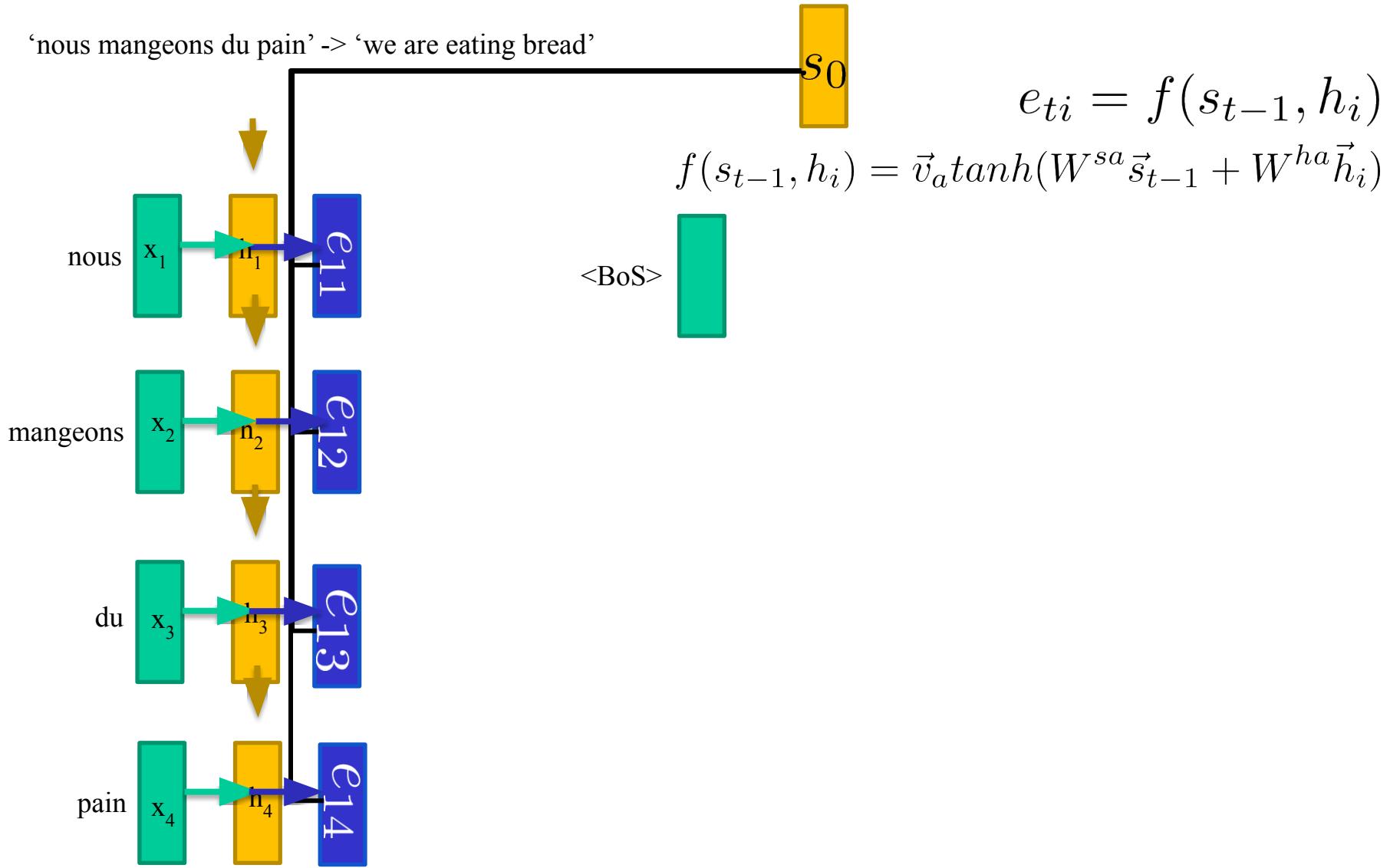
bread

Attention !

Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

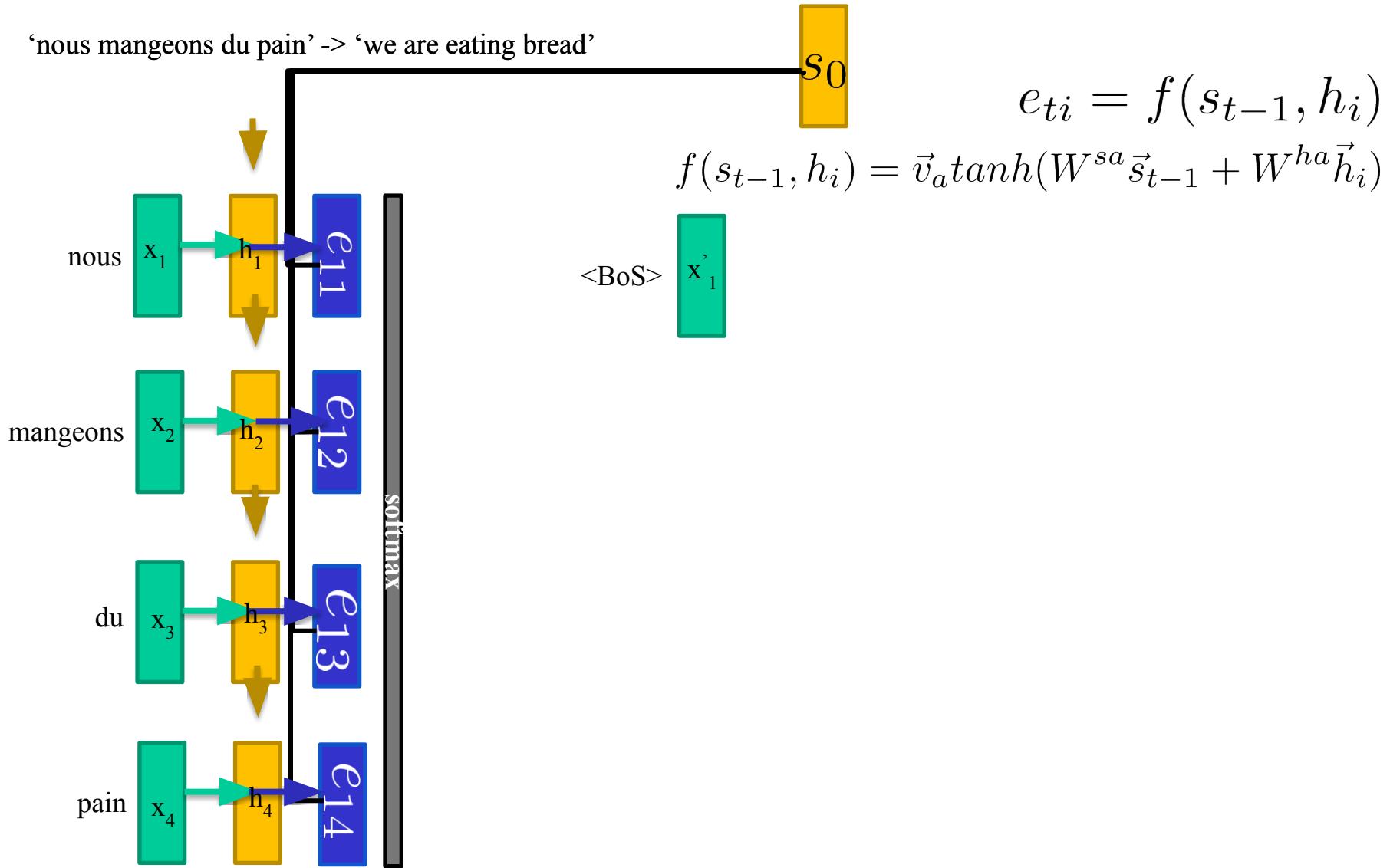
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

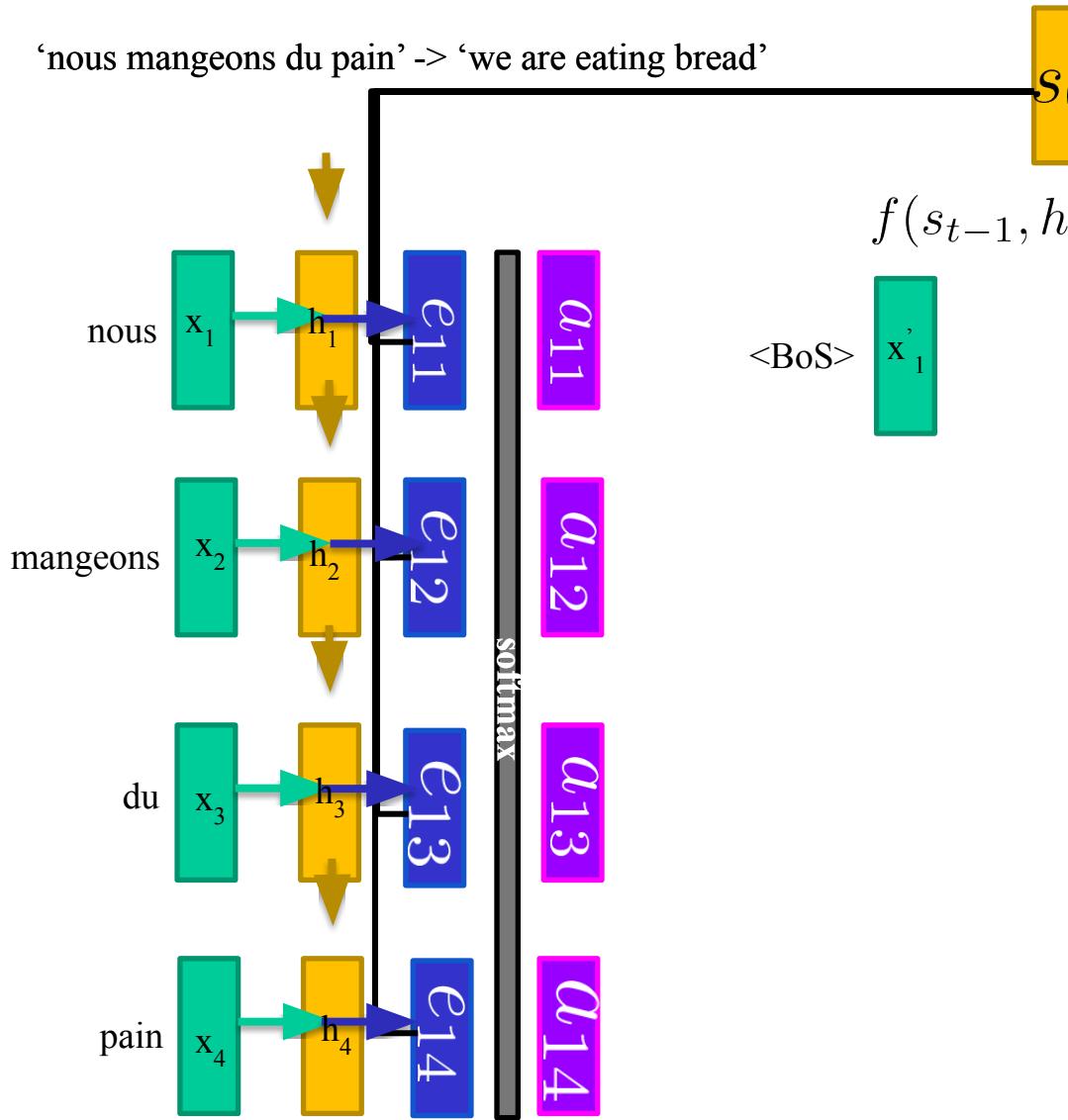
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

‘nous mangeons du pain’ -> ‘we are eating bread’



s_0

$$e_{ti} = f(s_{t-1}, h_i)$$

$$f(s_{t-1}, h_i) = \vec{v}_a \tanh(W^{sa} \vec{s}_{t-1} + W^{ha} \vec{h}_i)$$

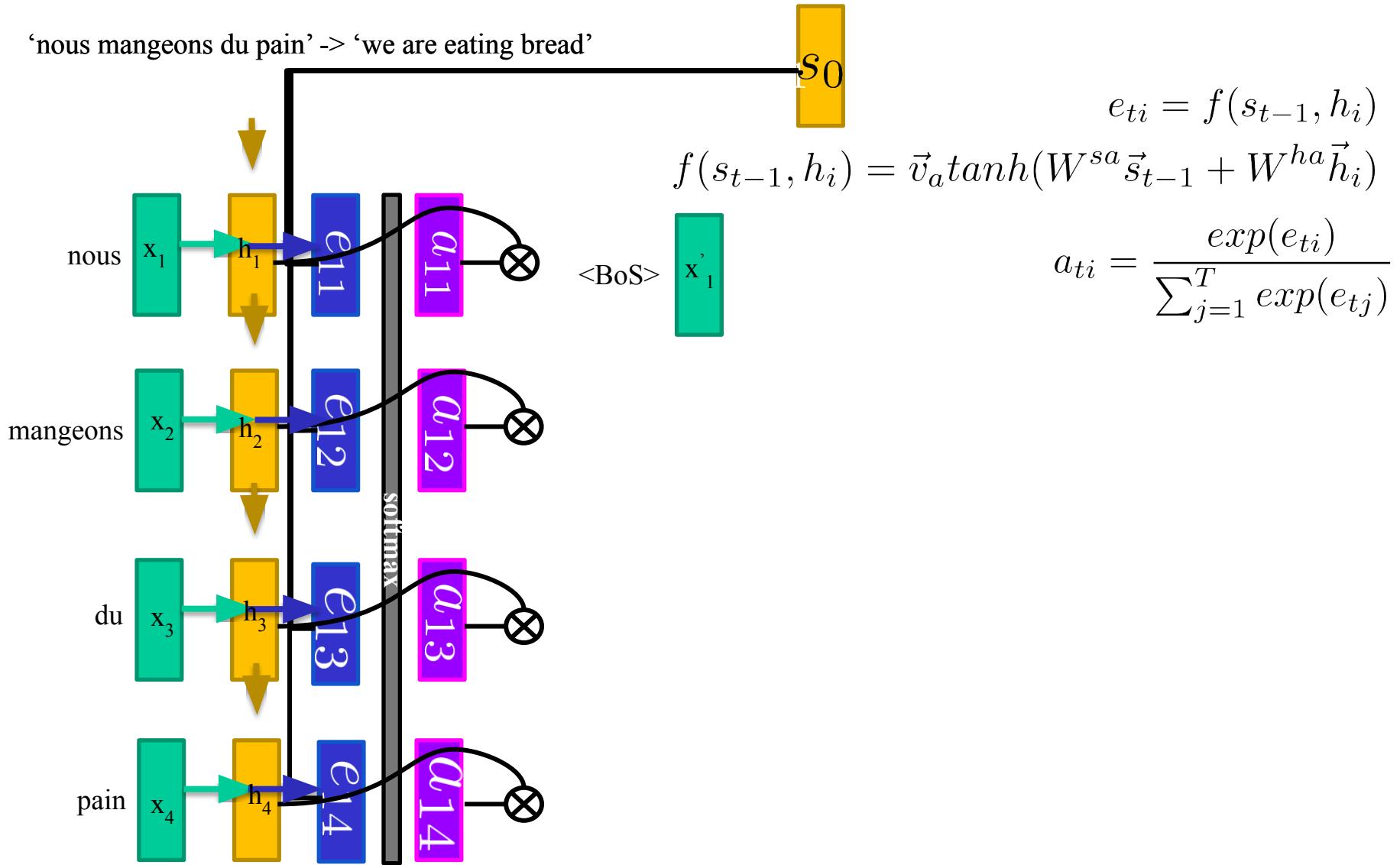
<BoS> x'_1

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

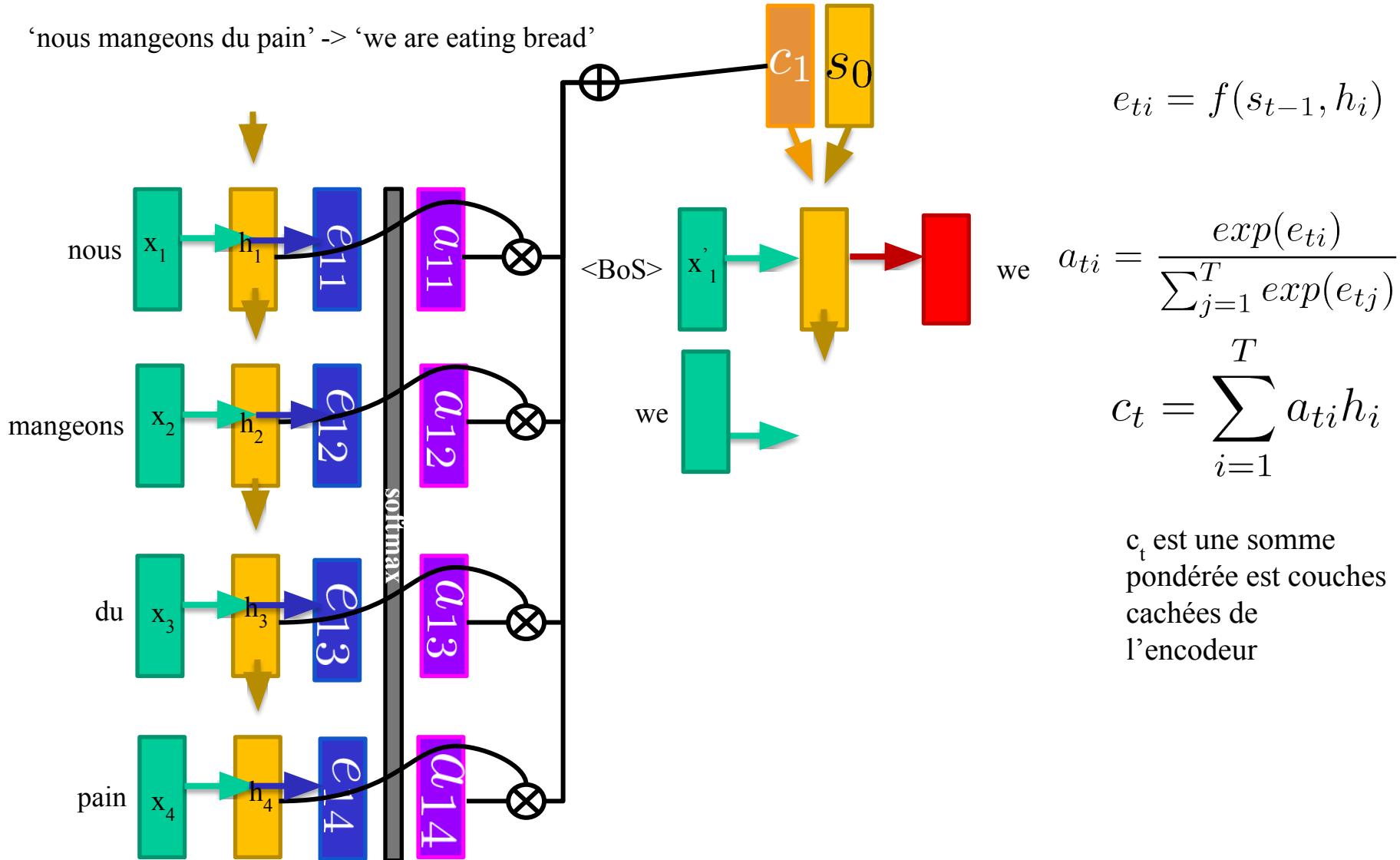
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

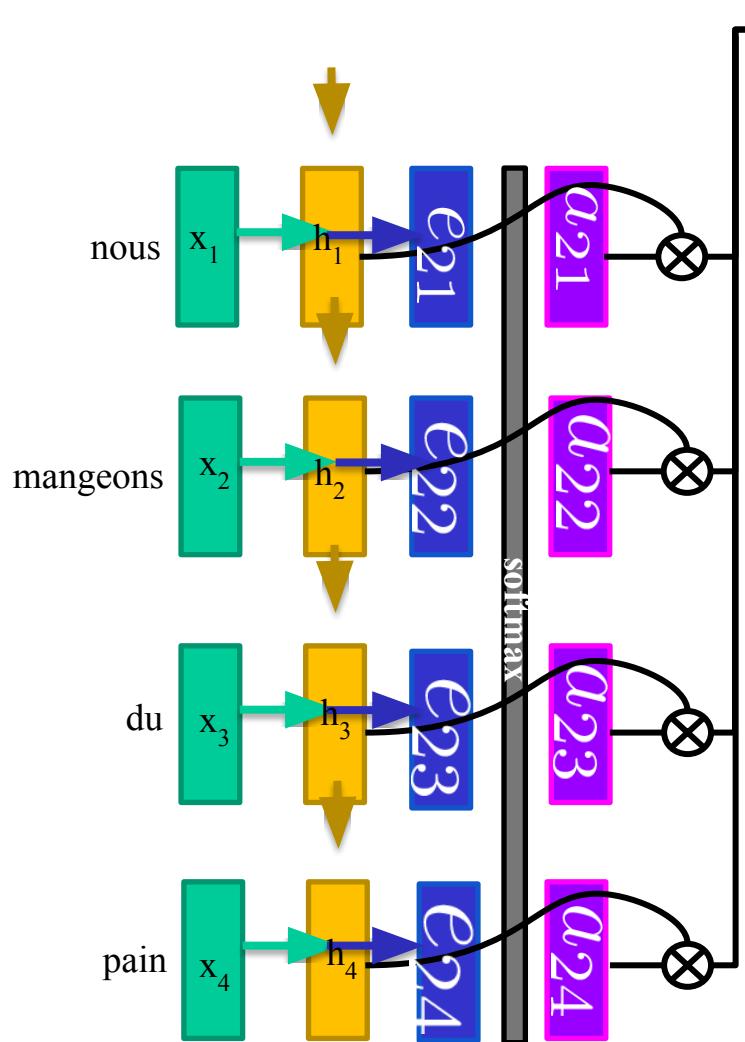
‘nous mangeons du pain’ -> ‘we are eating bread’



Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

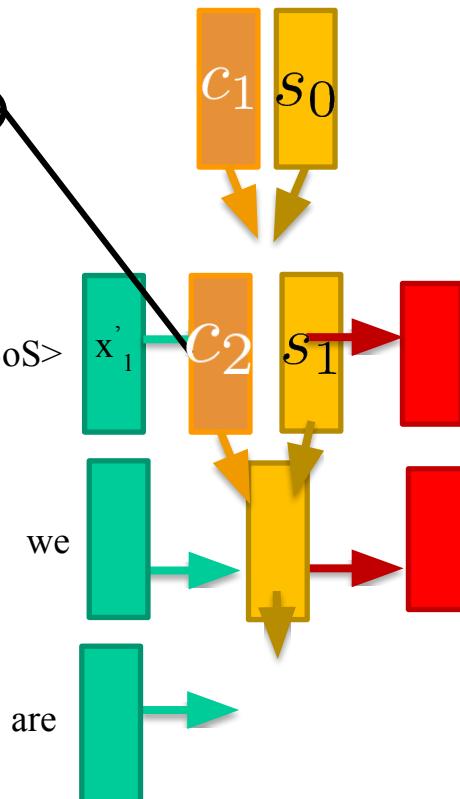
‘nous mangeons du pain’ -> ‘we are eating bread’



$$e_{ti} = f(s_{t-1}, h_i)$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_{i=1}^T a_{ti} h_i$$

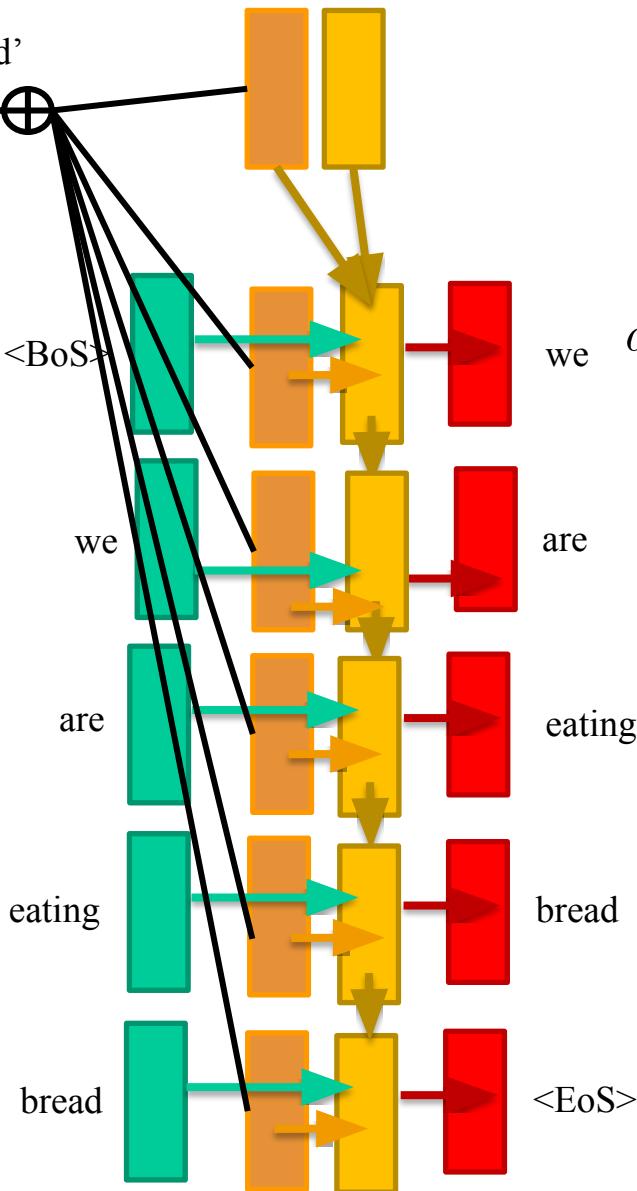
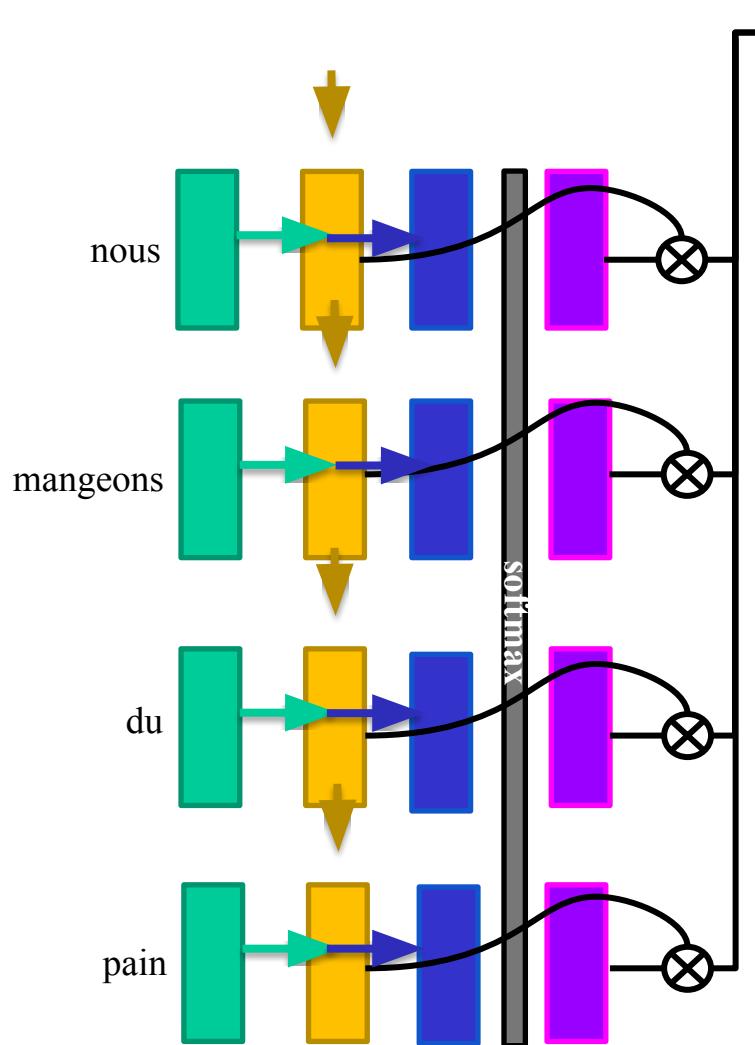


c_t est une somme pondérée est couches cachées de l’encodeur

Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

‘nous mangeons du pain’ -> ‘we are eating bread’



$$e_{ti} = f(s_{t-1}, h_i)$$

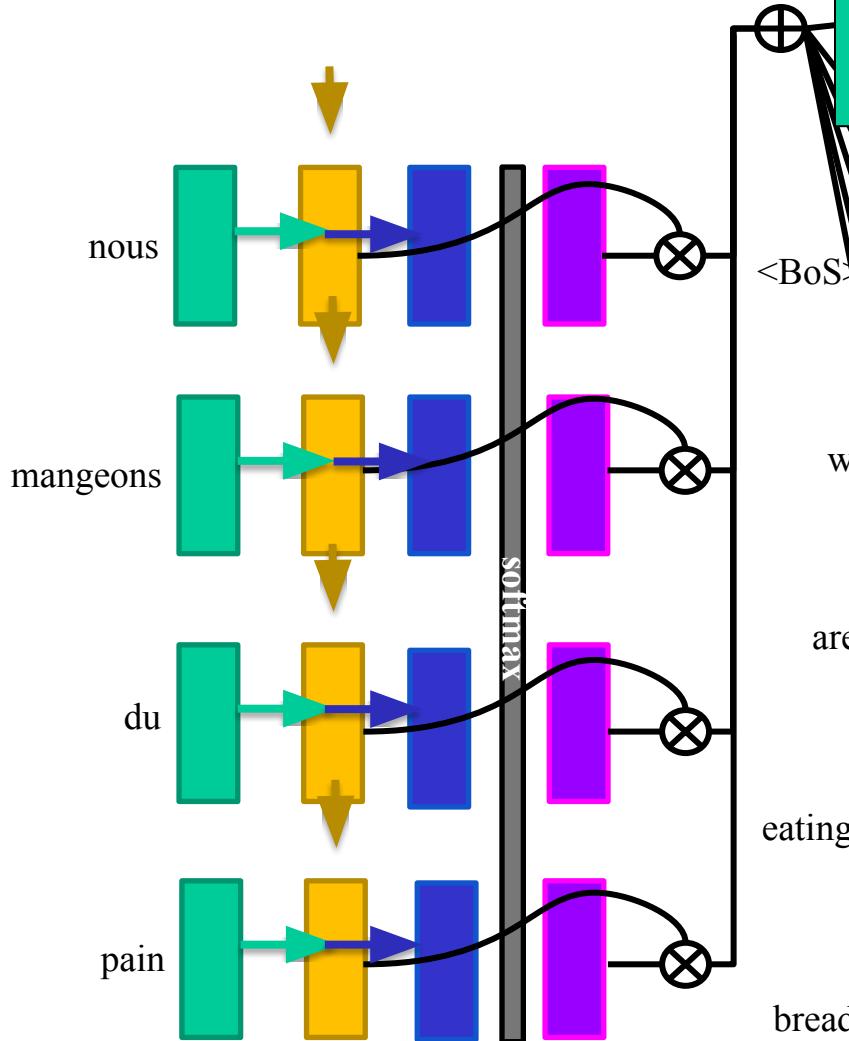
$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_{i=1}^T a_{ti} h_i$$

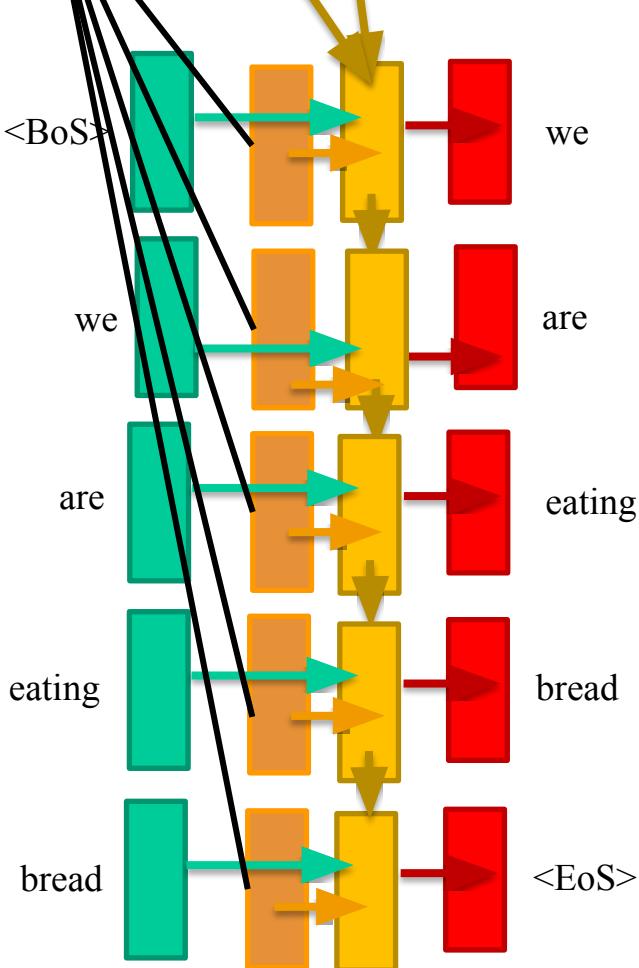
Seq2Seq avec attention

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

‘nous mangeons du pain’ -> ‘we are eating bread’



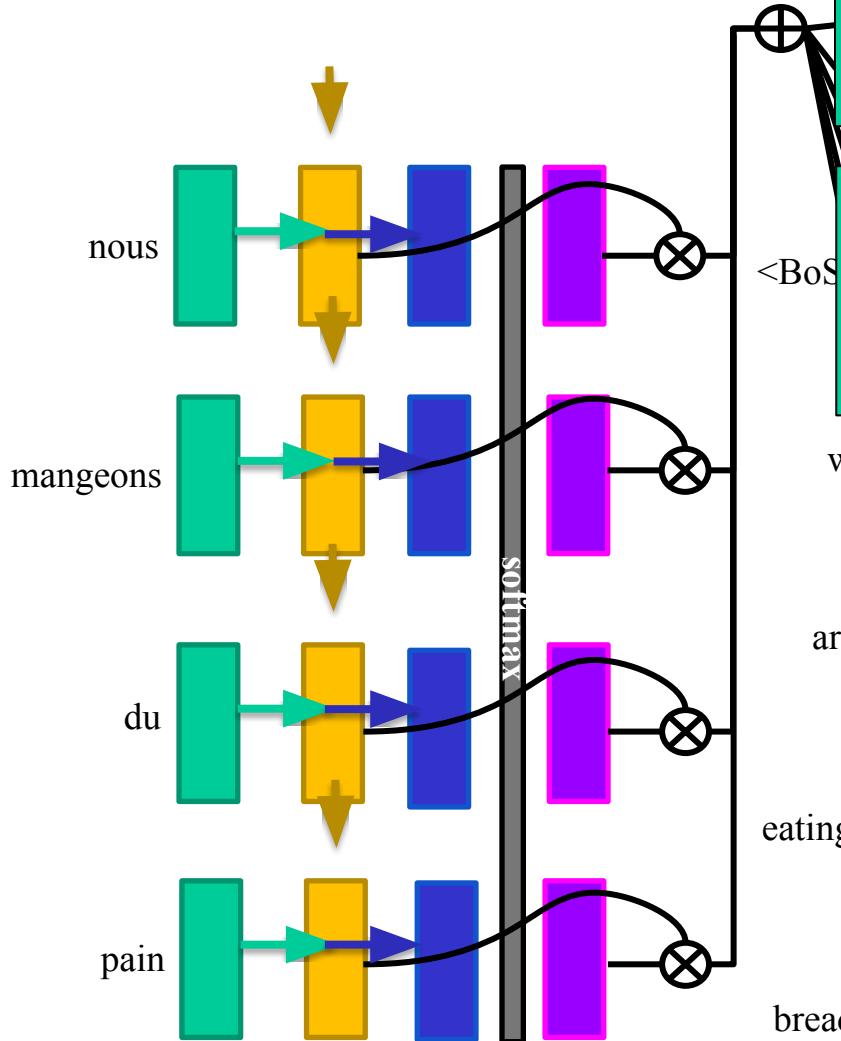
Le modèle apprend par lui-même à “porter attention” à certain mots de l’encodeur à chaque “timestep” du décodage



Seq2Seq avec attention

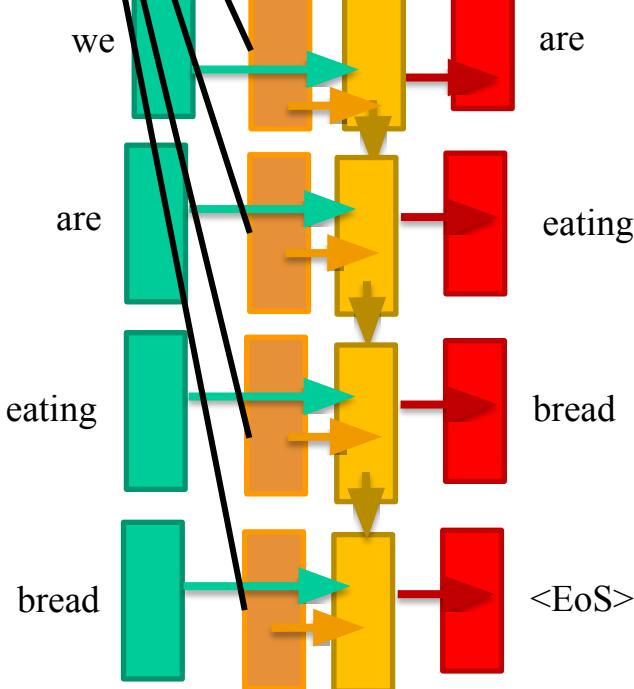
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

‘nous mangeons du pain’ -> ‘we are eating bread’

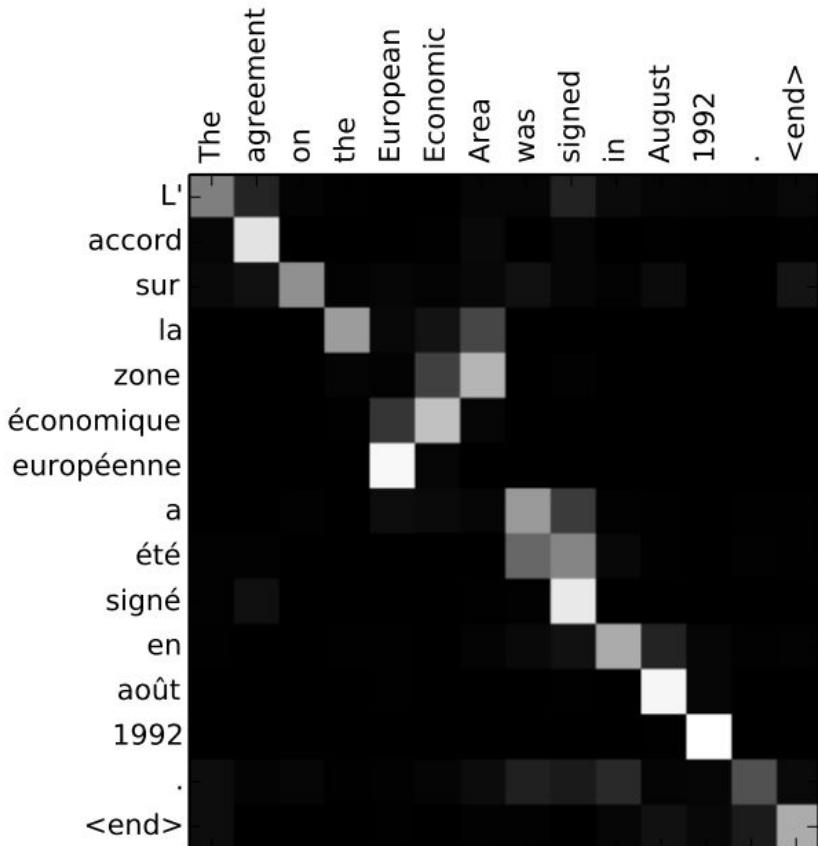


Le modèle apprend par lui-même à “porter attention” à certain mots de l’encodeur à chaque “timestep” du décodage

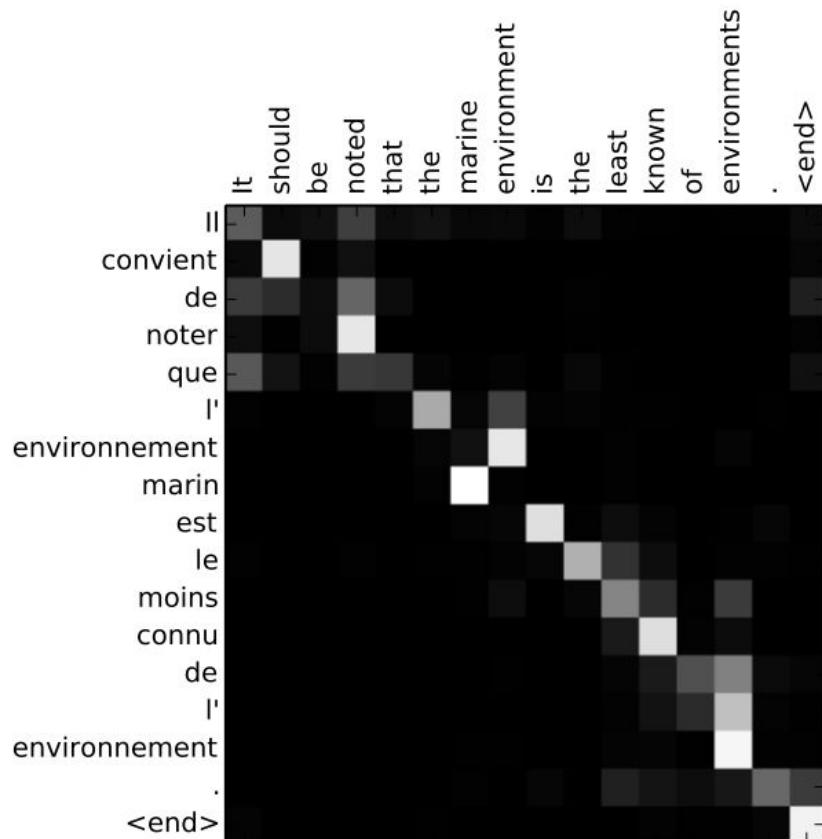
Rien n'est “forcé”, tout est différentiable, donc appris par le réseau !



Seq2Seq avec attention



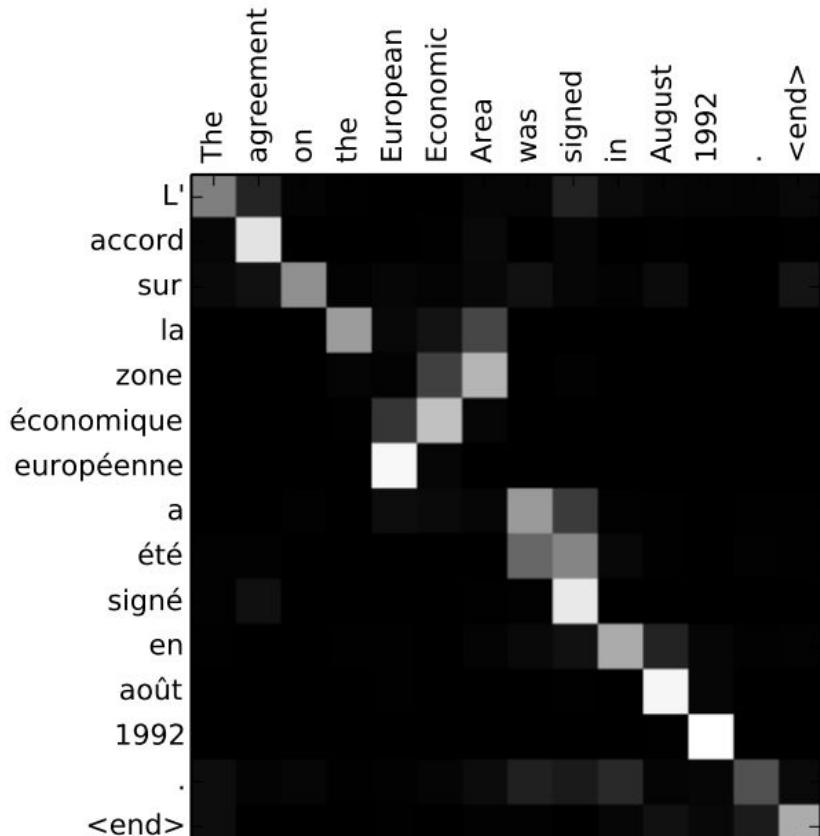
(a)



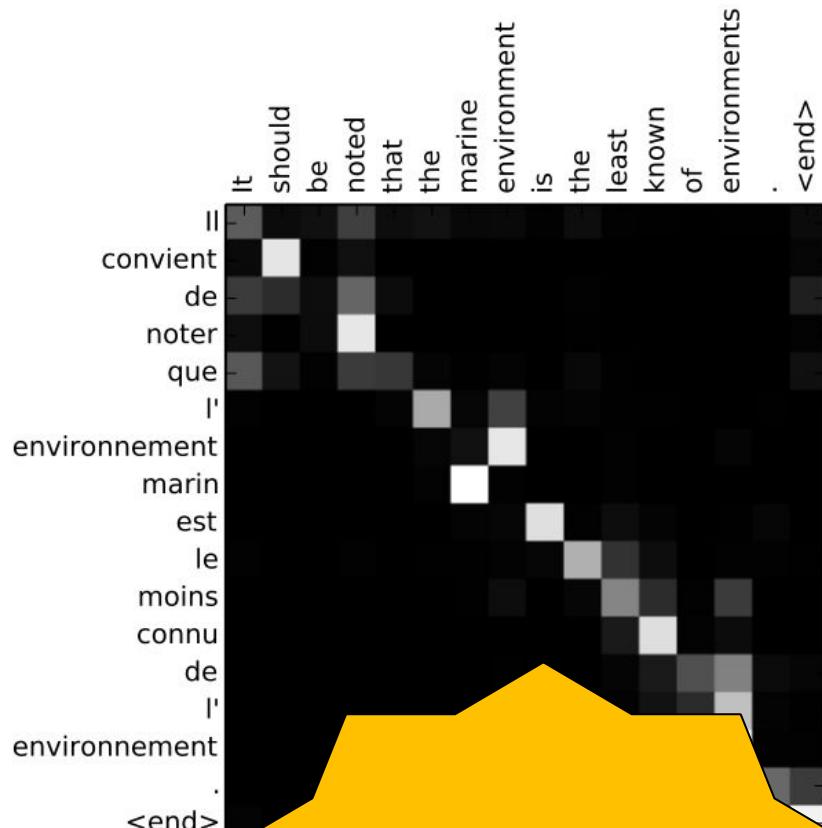
(b)

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Seq2Seq avec attention



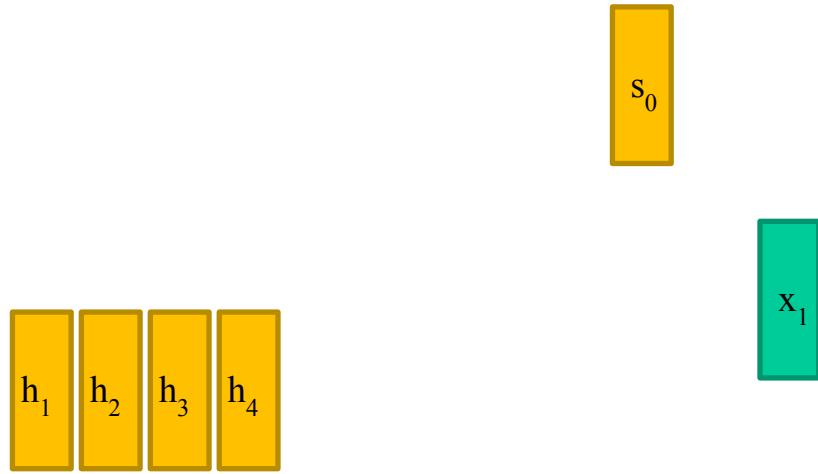
(a)



**Ajoute de
l'interprétabilité
au modèle !**

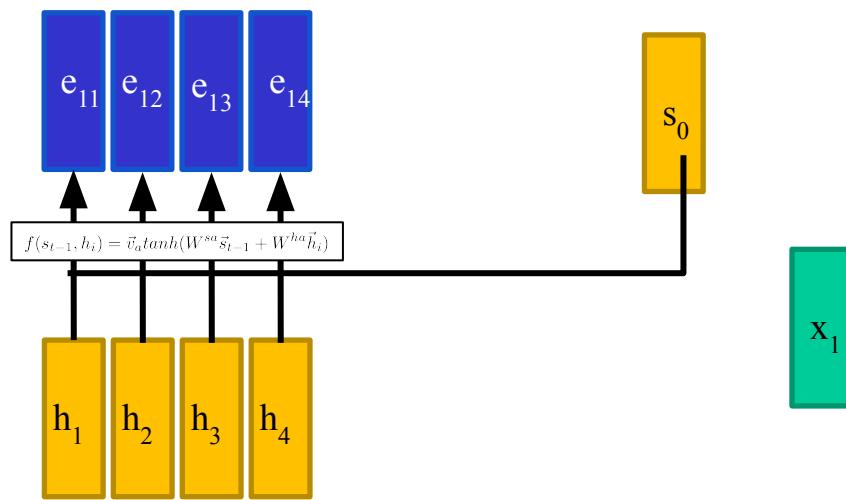
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

Attention: version simplifiée

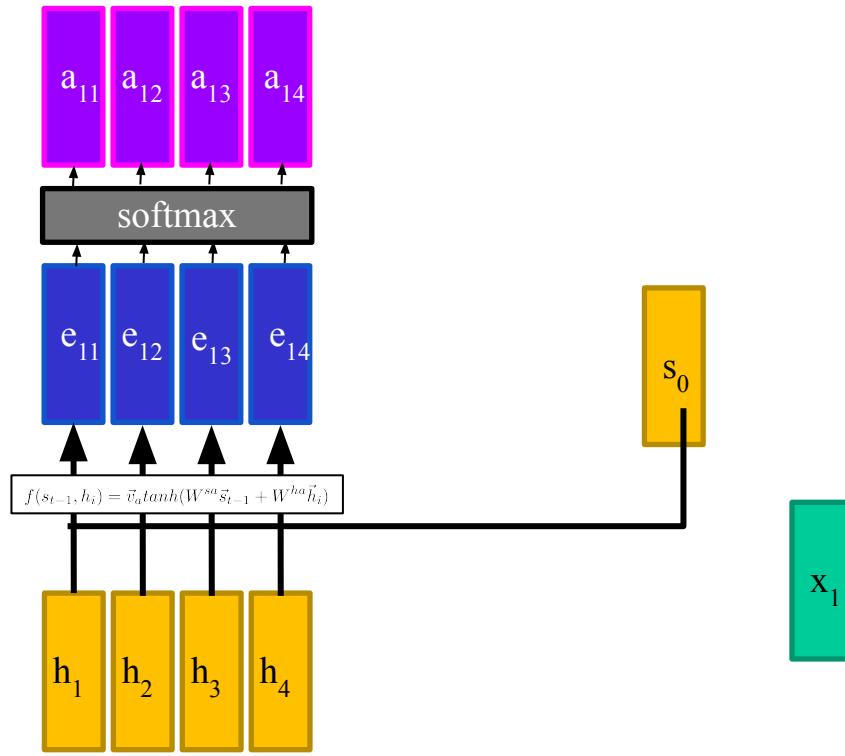


Attention: version simplifiée

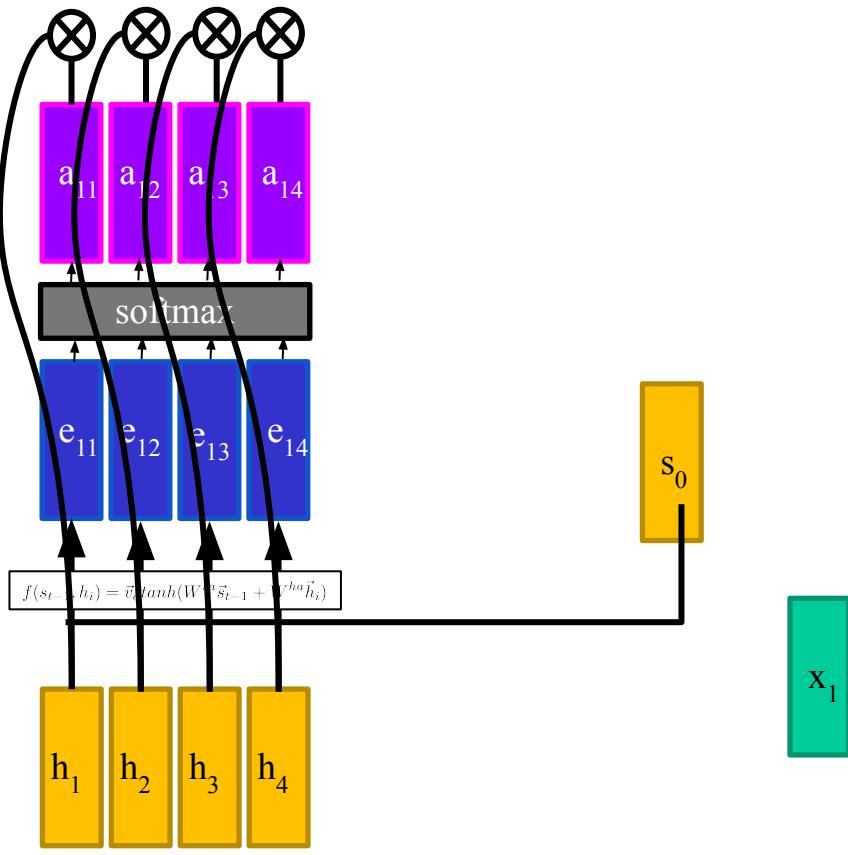
$$f(s_{t-1}, h_i) = \vec{v}_a \tanh(W^{sa} \vec{s}_{t-1} + W^{ha} \vec{h}_i)$$



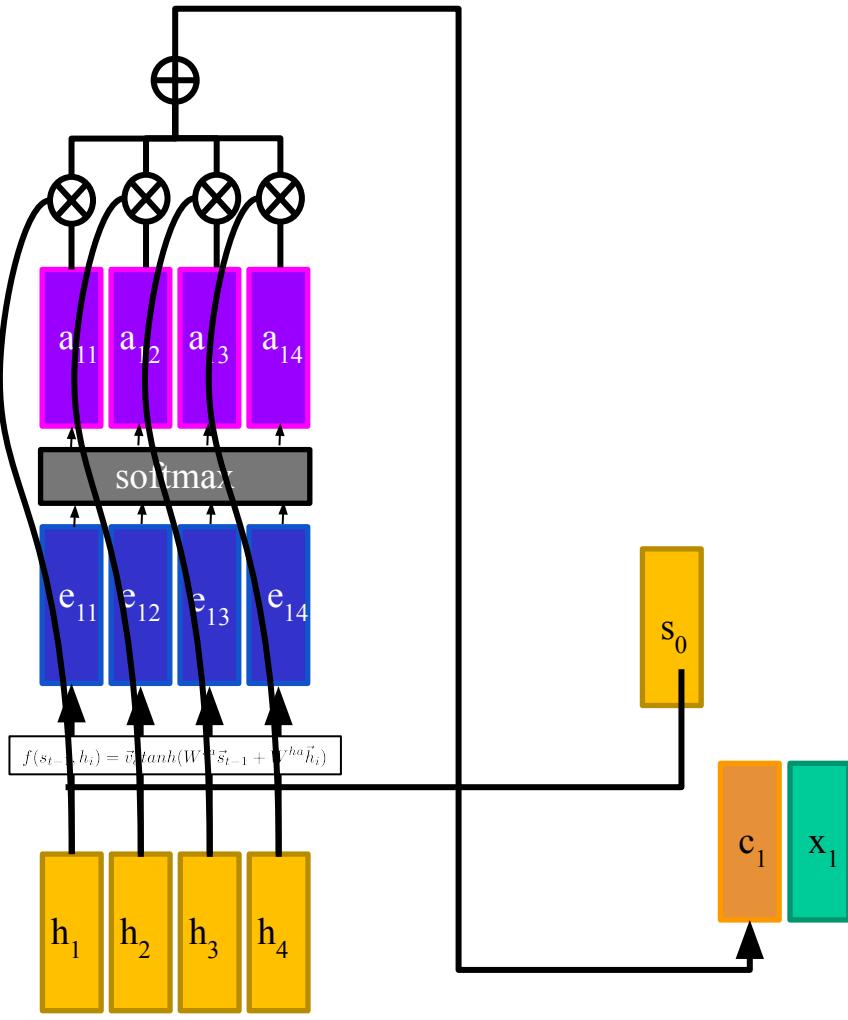
Attention: version simplifiée



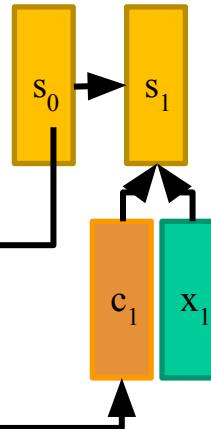
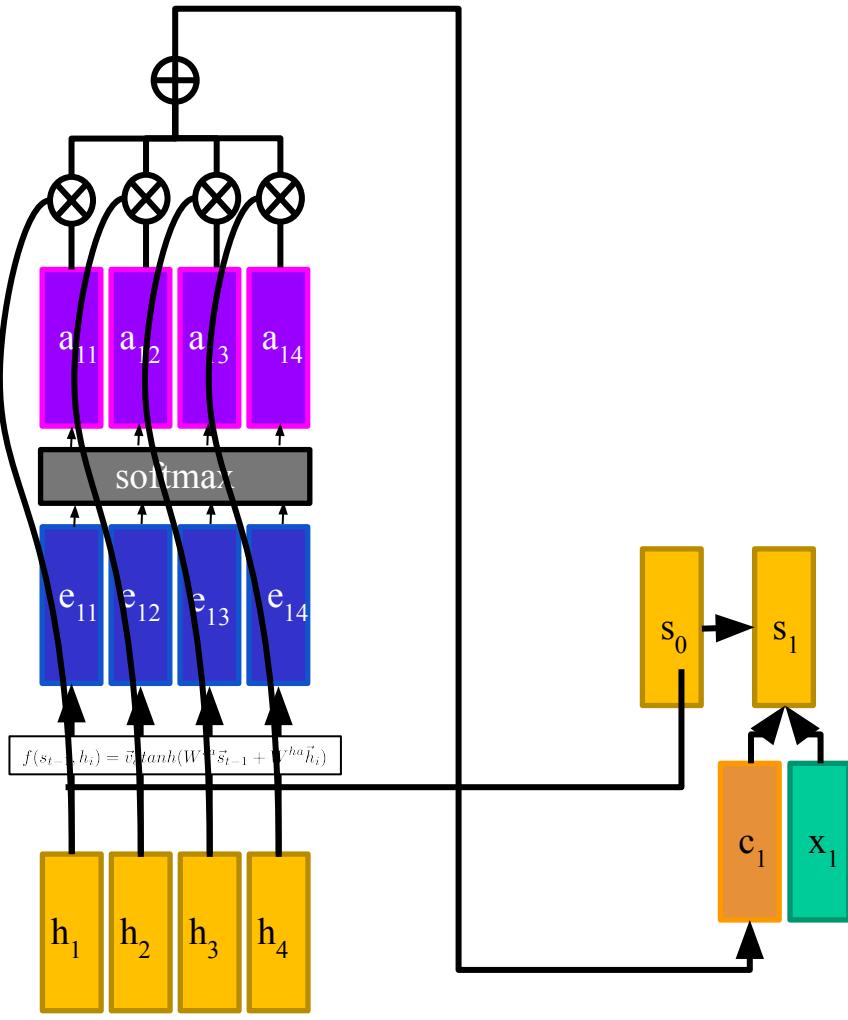
Attention: version simplifiée



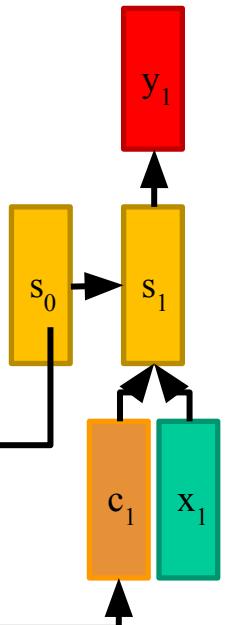
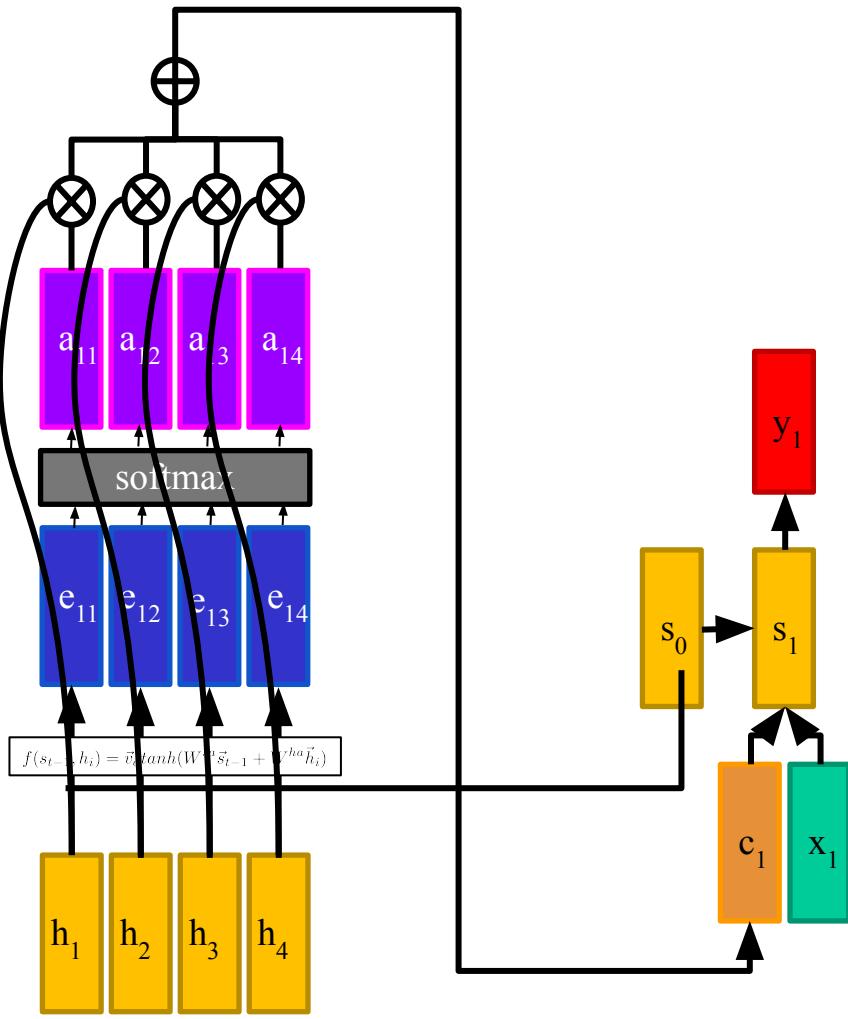
Attention: version simplifiée



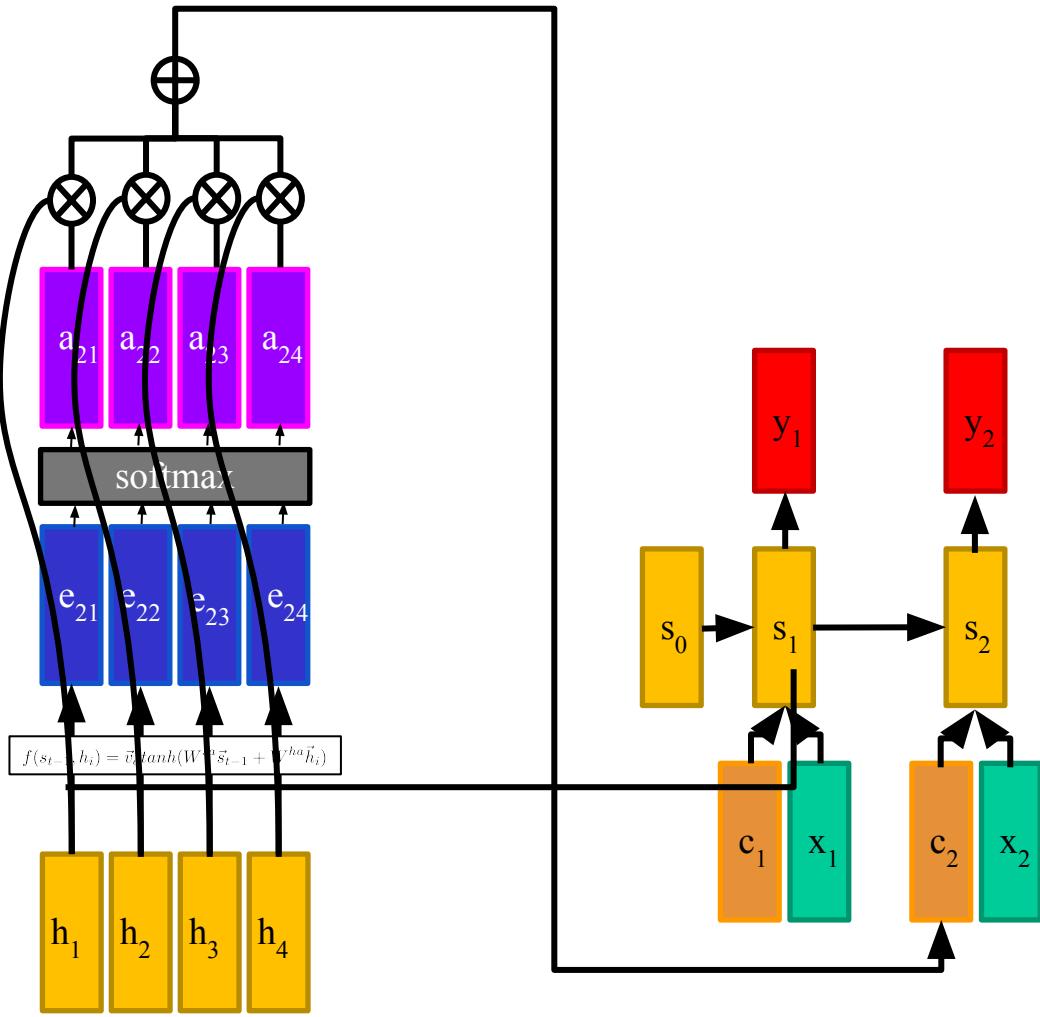
Attention: version simplifiée



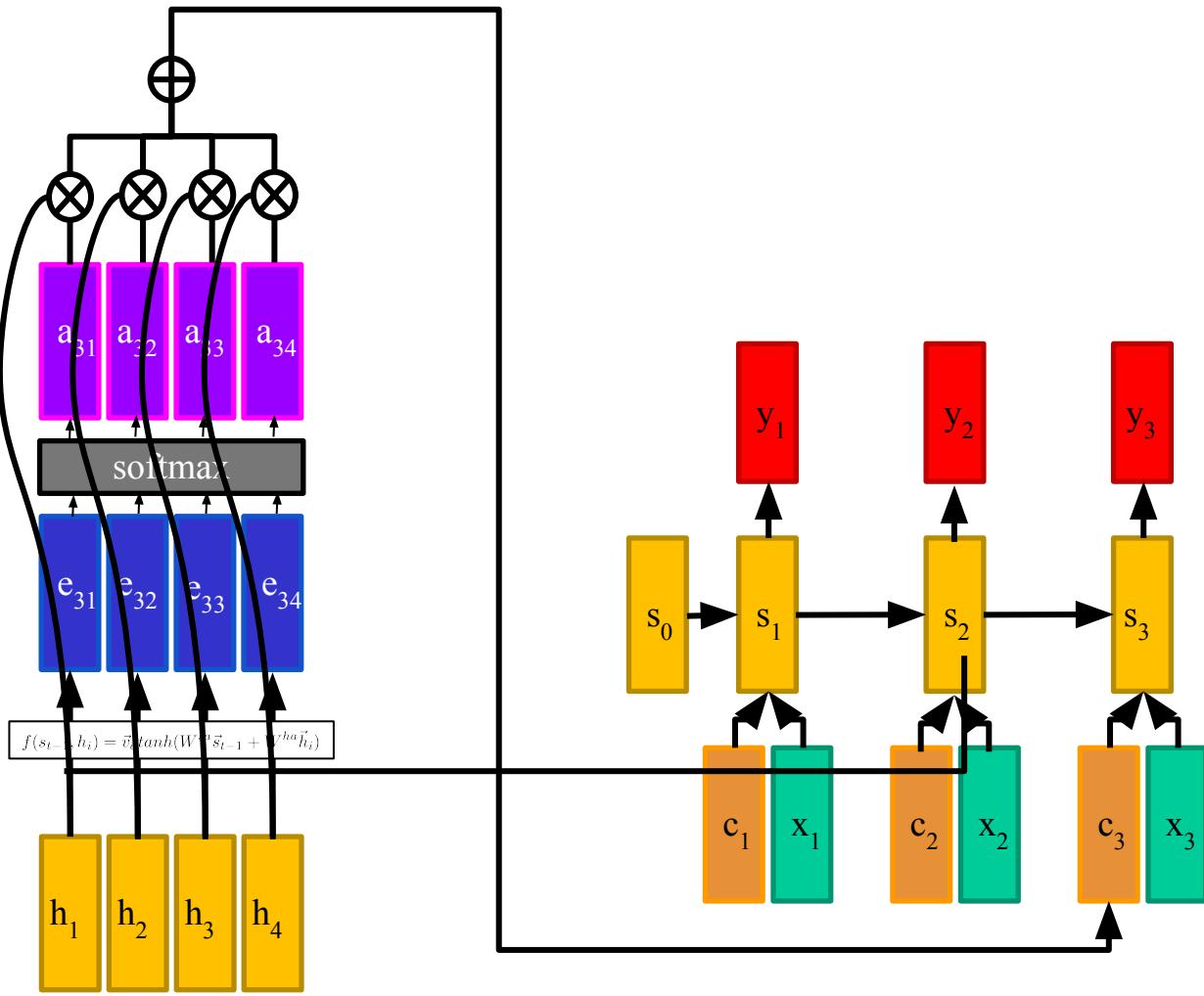
Attention: version simplifiée



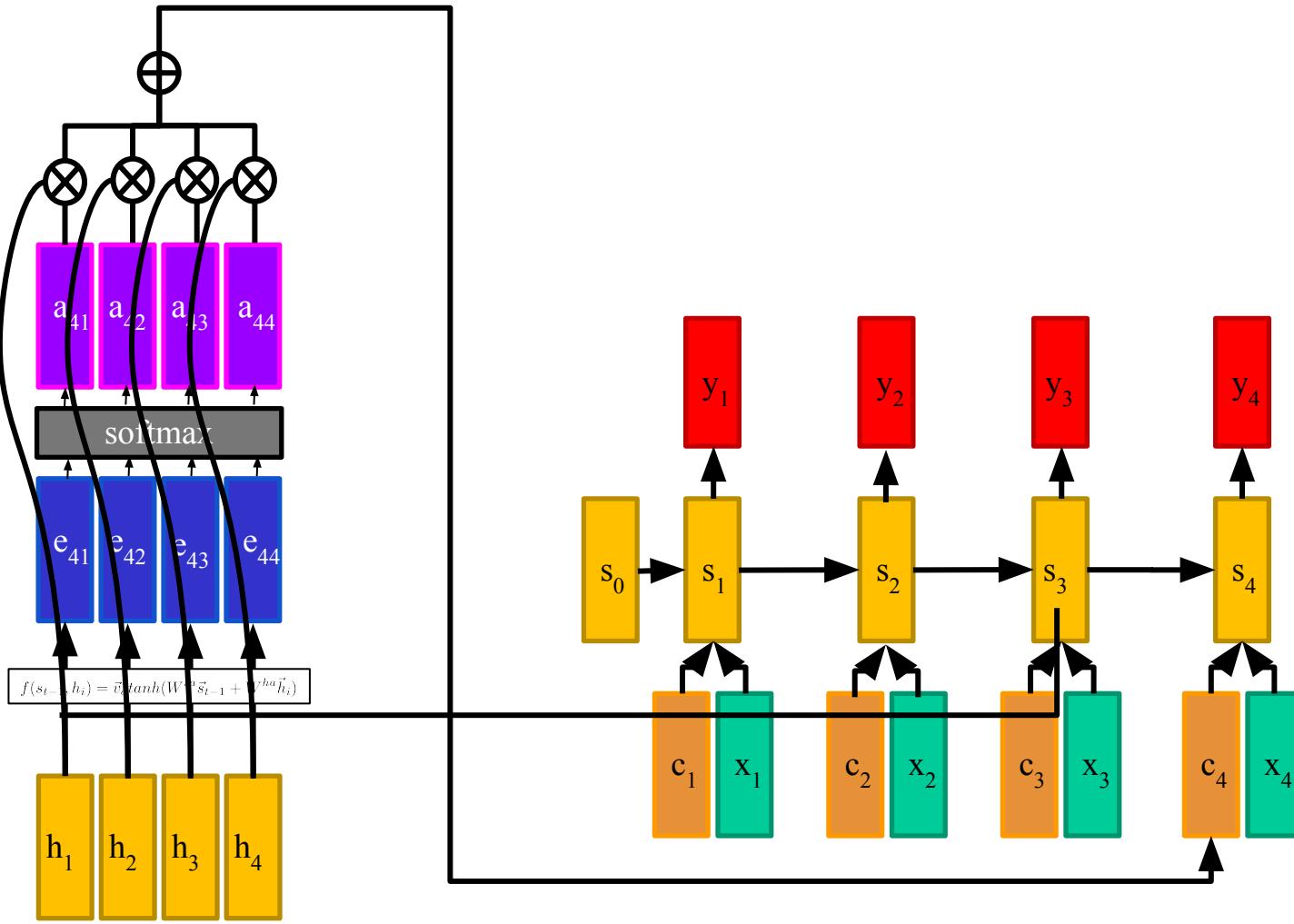
Attention: version simplifiée



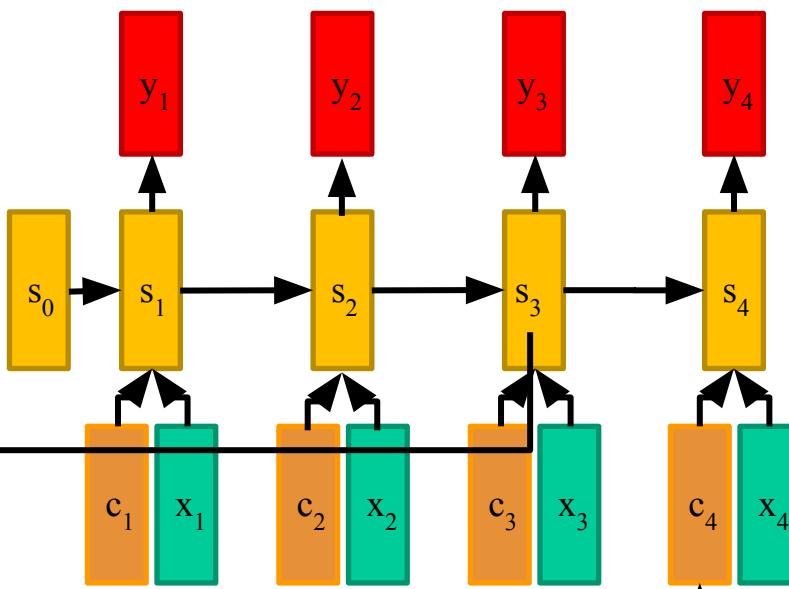
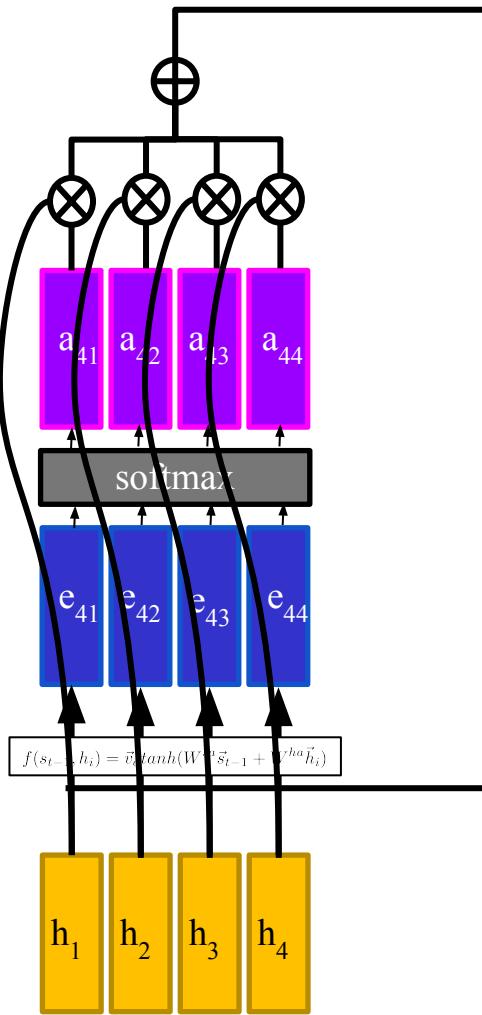
Attention: version simplifiée



Attention: version simplifiée



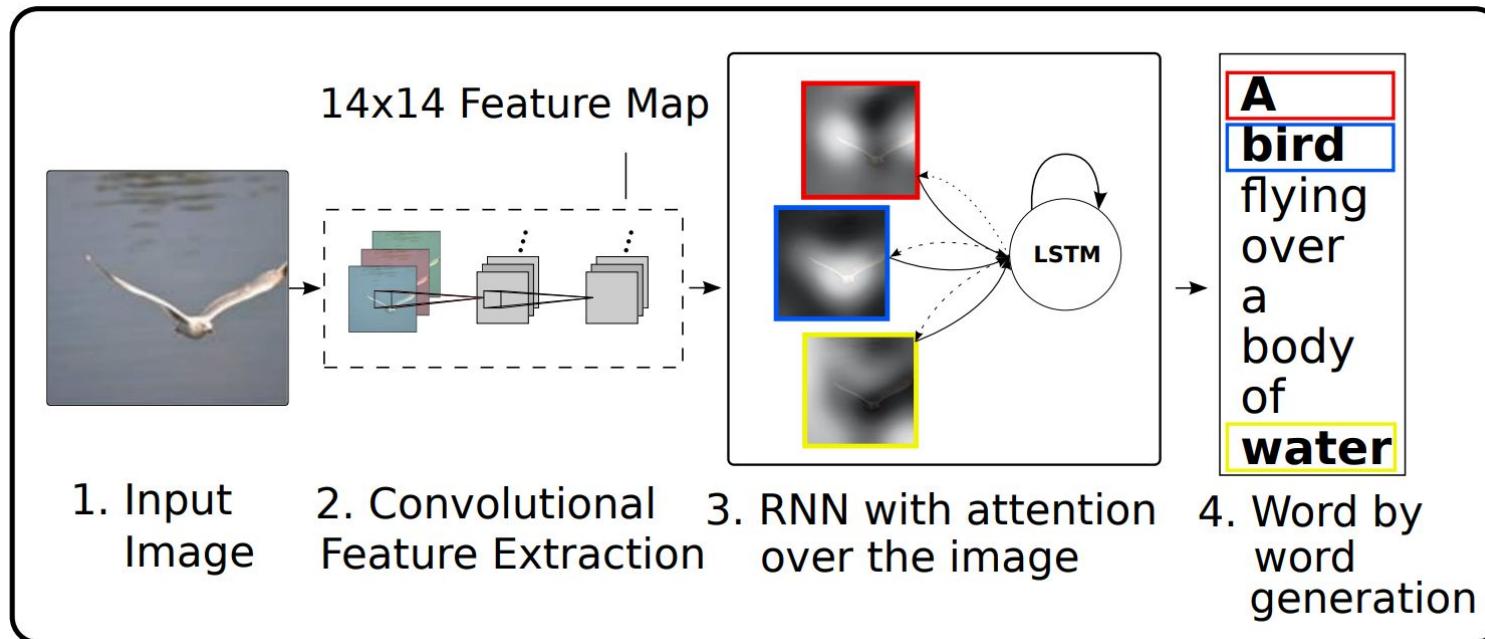
Attention: version simplifiée



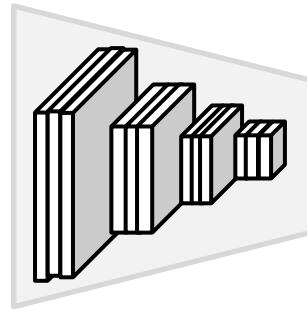
Forcément sur
des séquences ?
Non !
L'encodeur est
fixe !

Modèles d'attention

Réseau récurrent pour du captioning capable de « concentrer son attention » sur les zones de l'image associés aux mots.

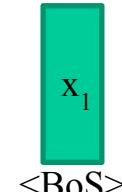


Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR.

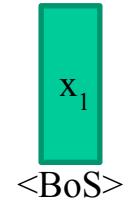
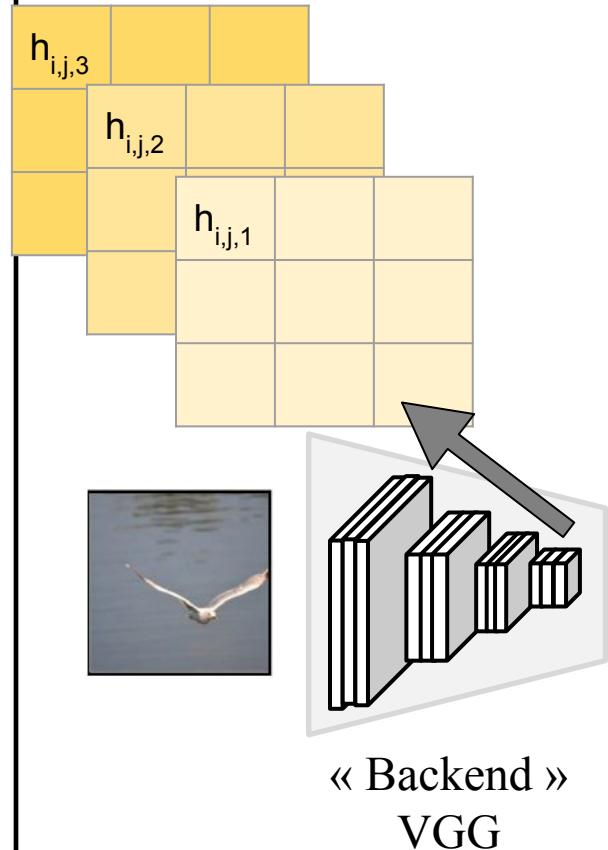


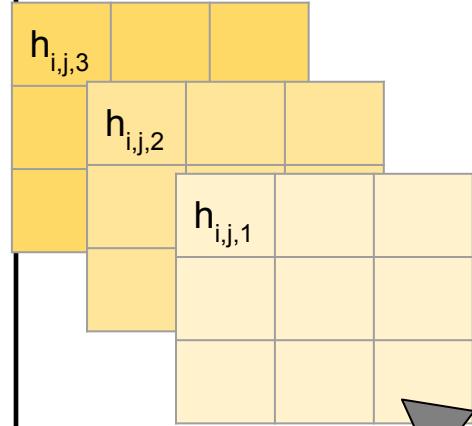
« Backend »

VGG

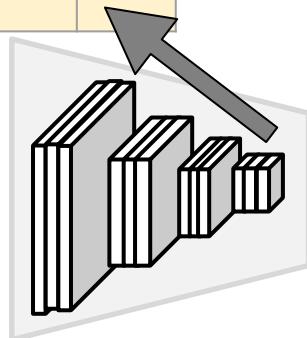


<BoS>

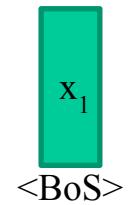


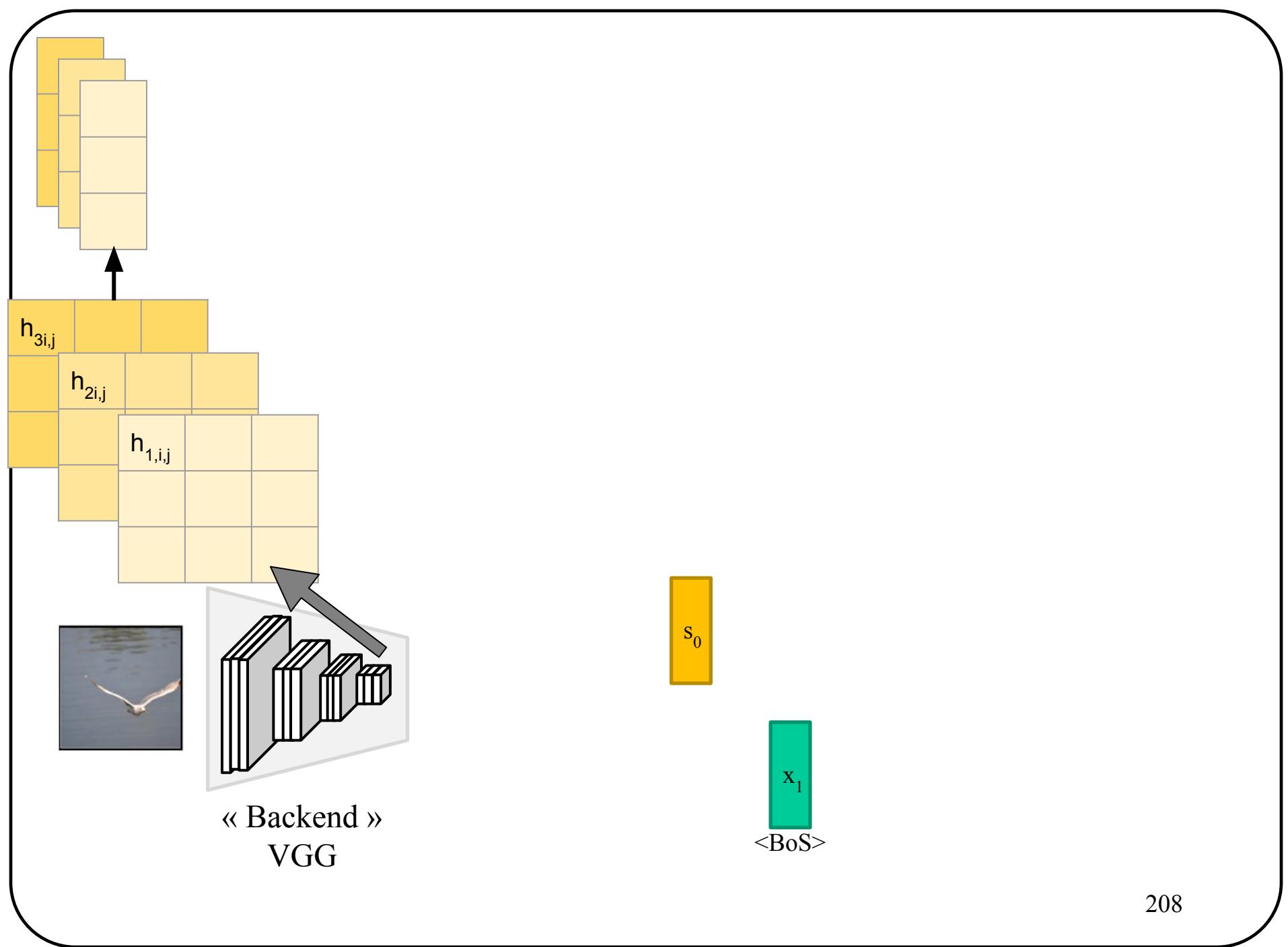


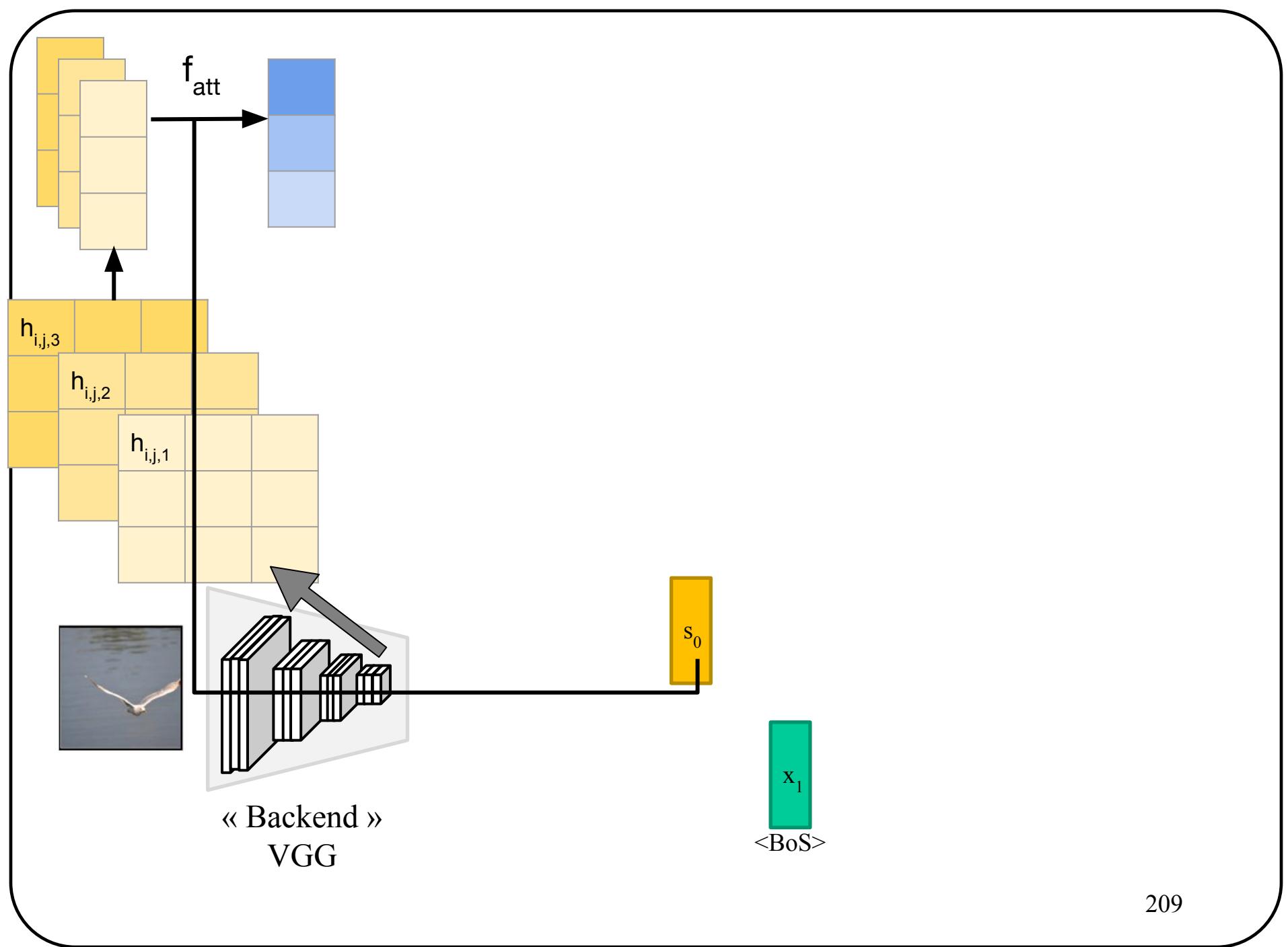
Les positions i, j correspondent à $h_{1,2,3\dots}$
 Les filtres correspondent aux composantes de h

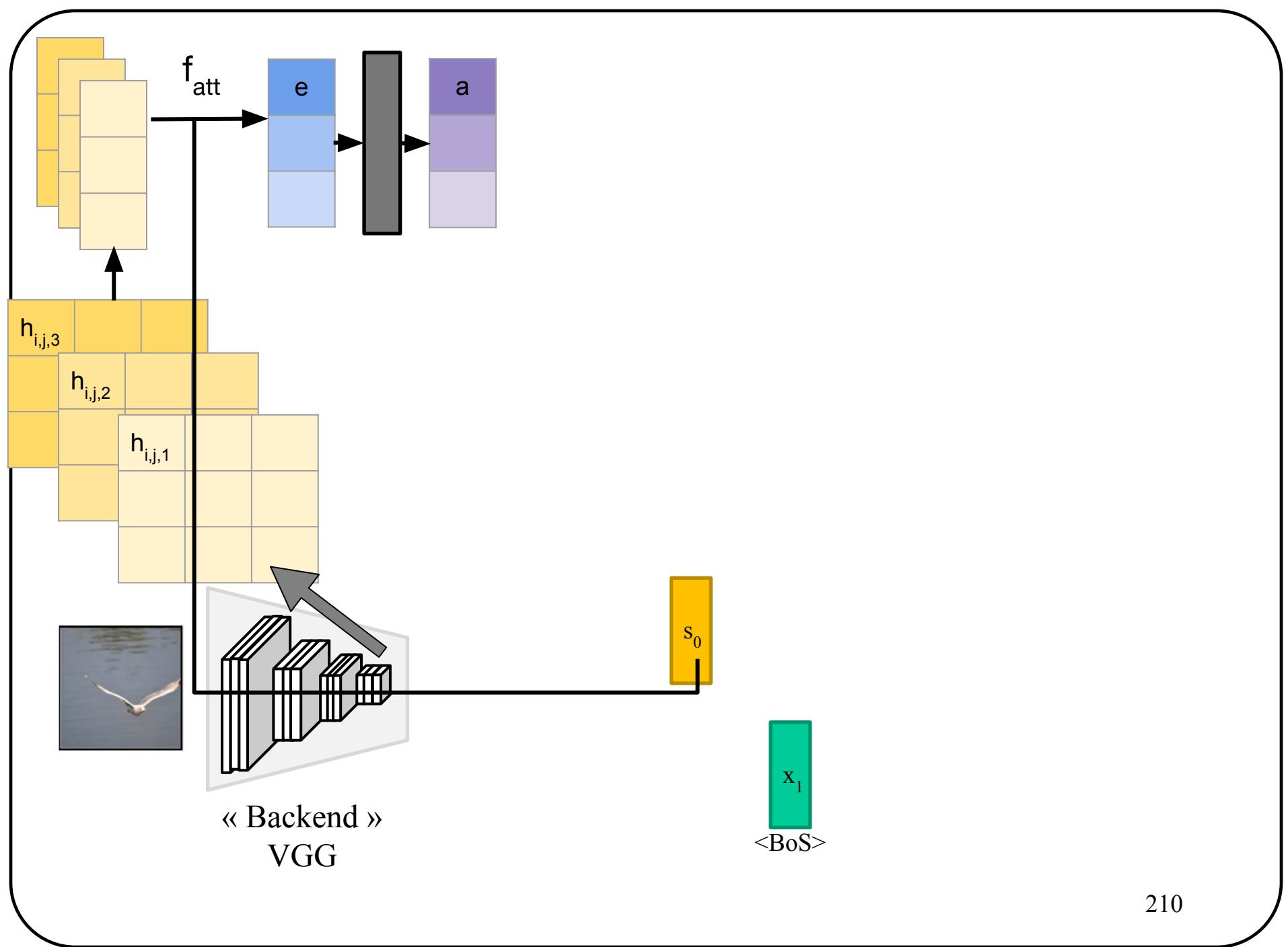


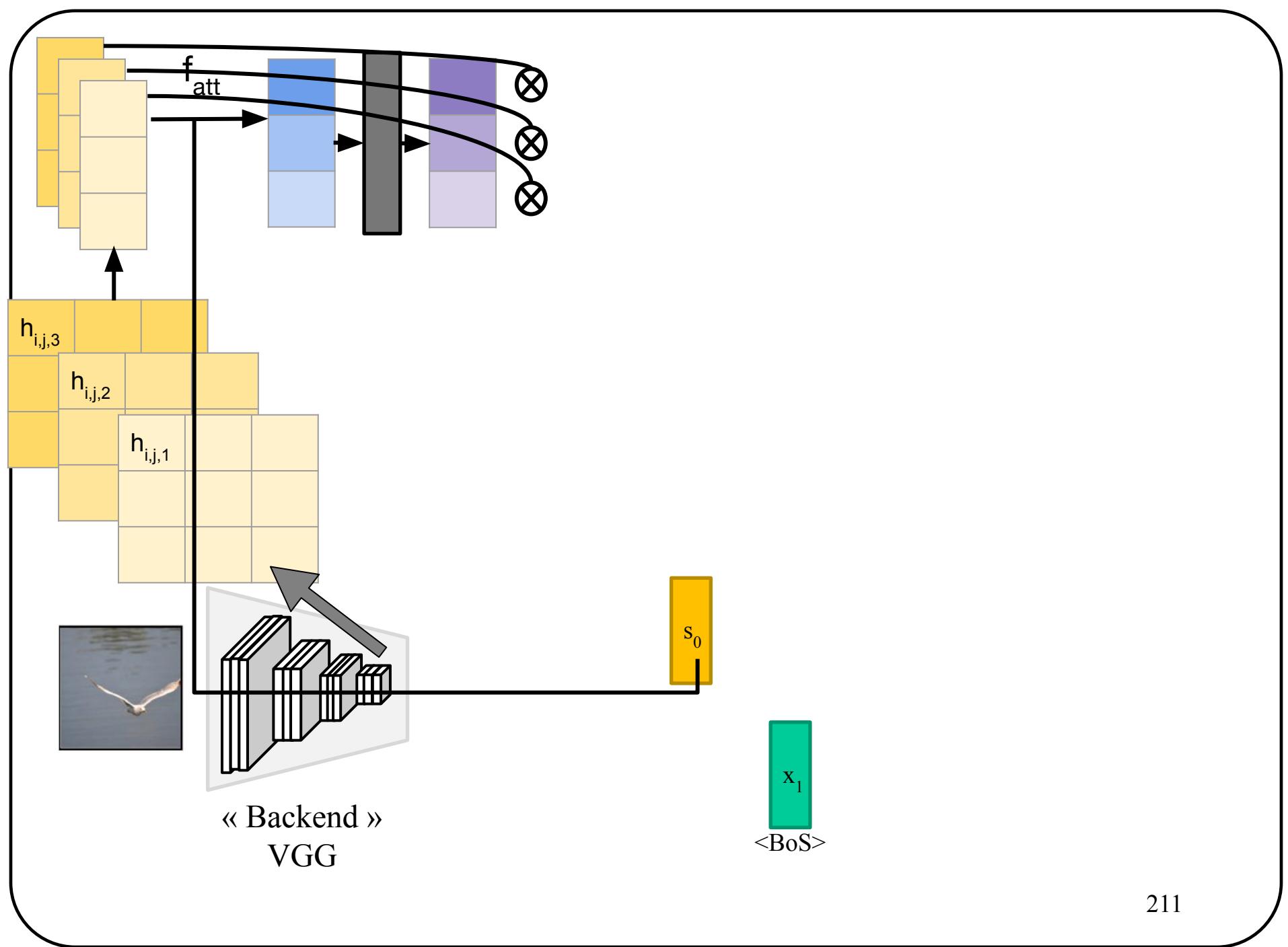
« Backend »
 VGG

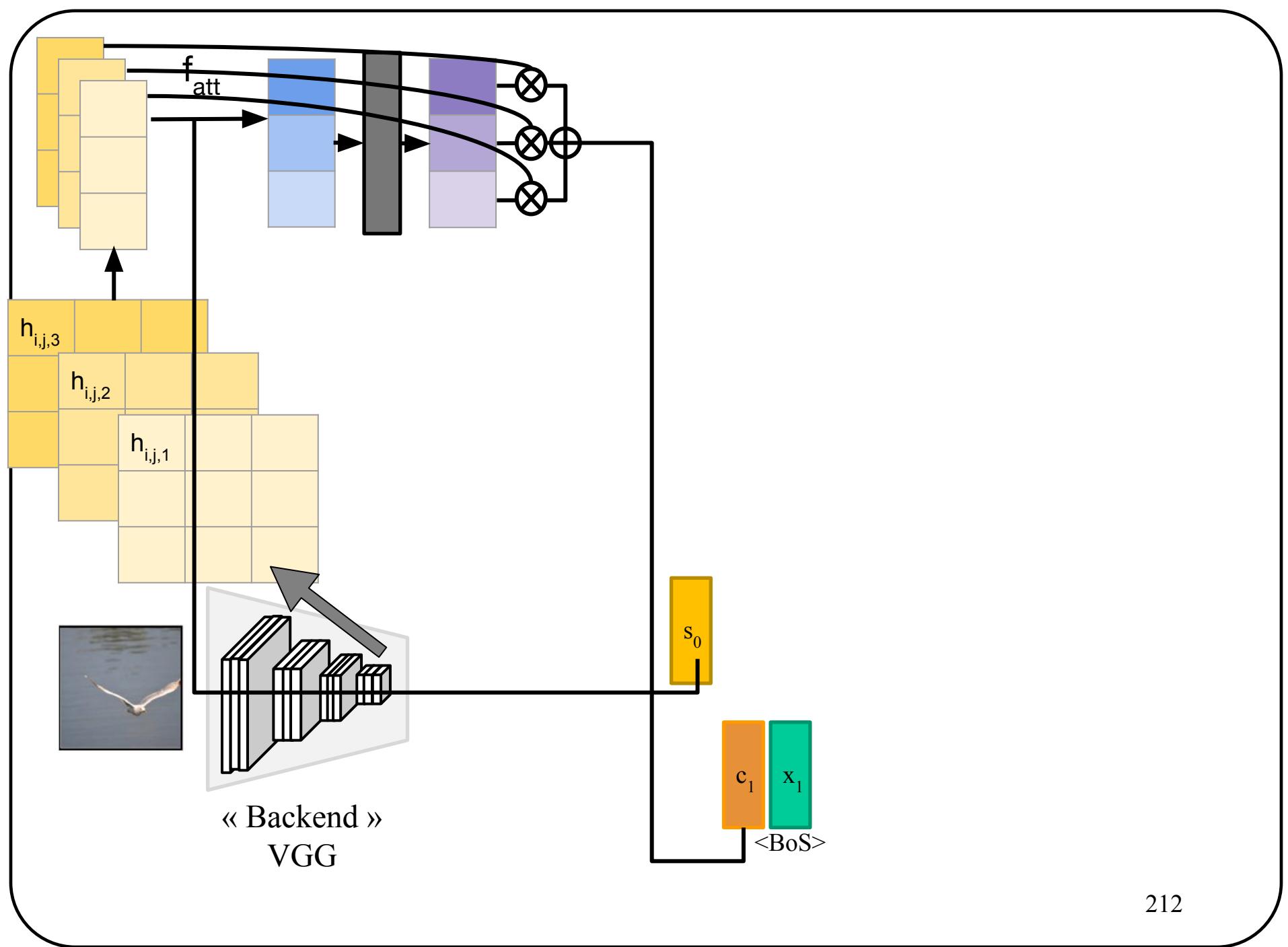


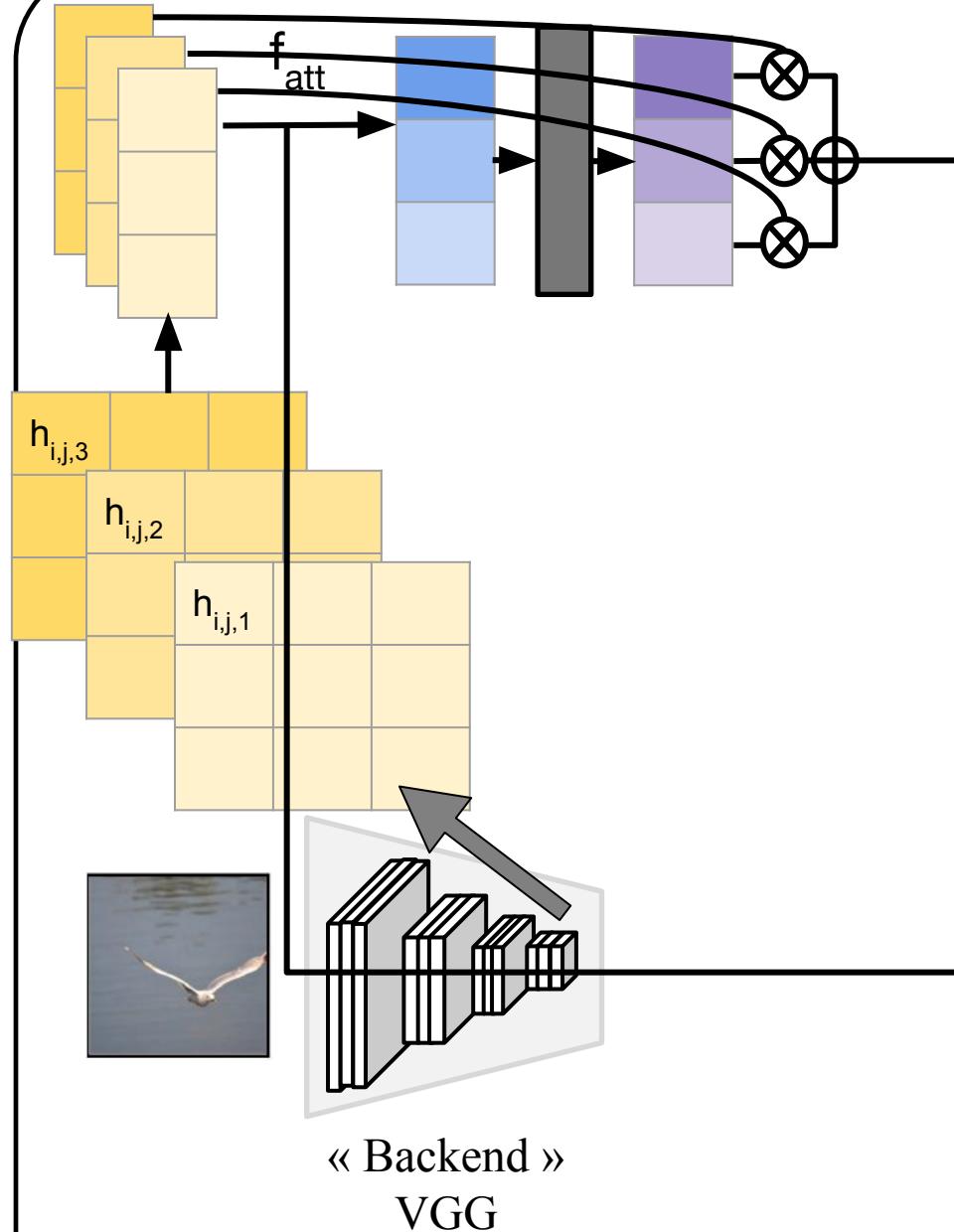




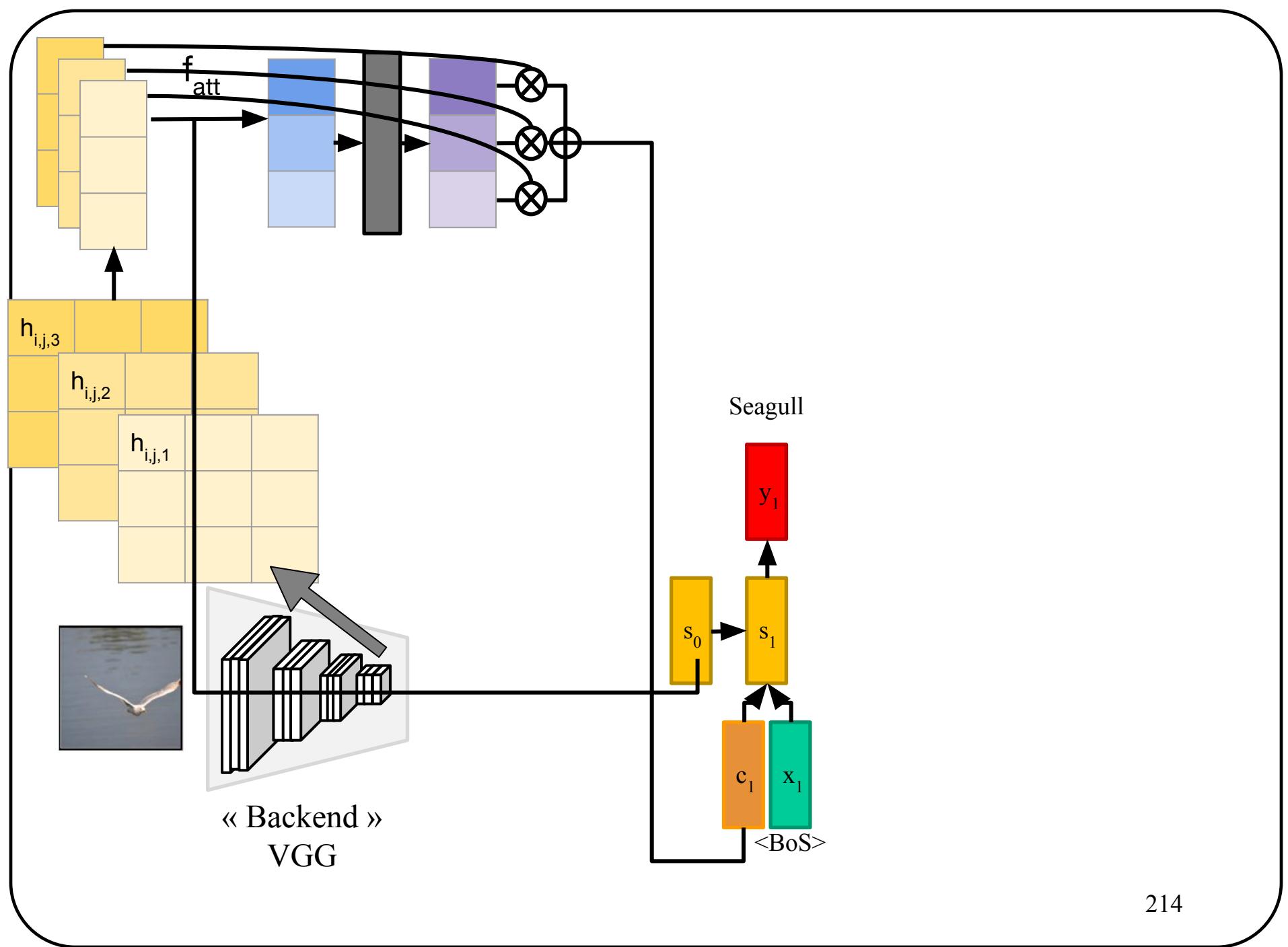


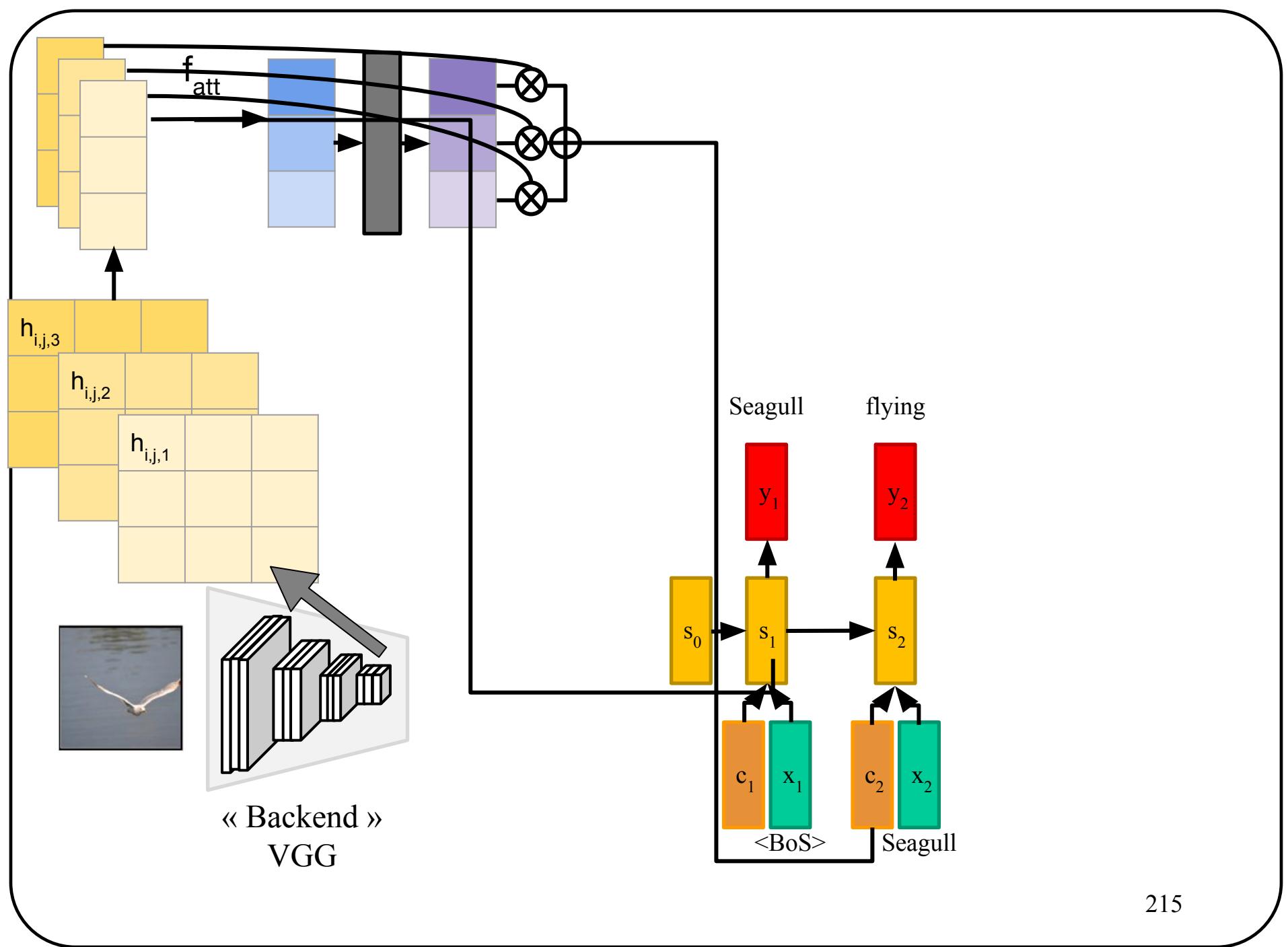


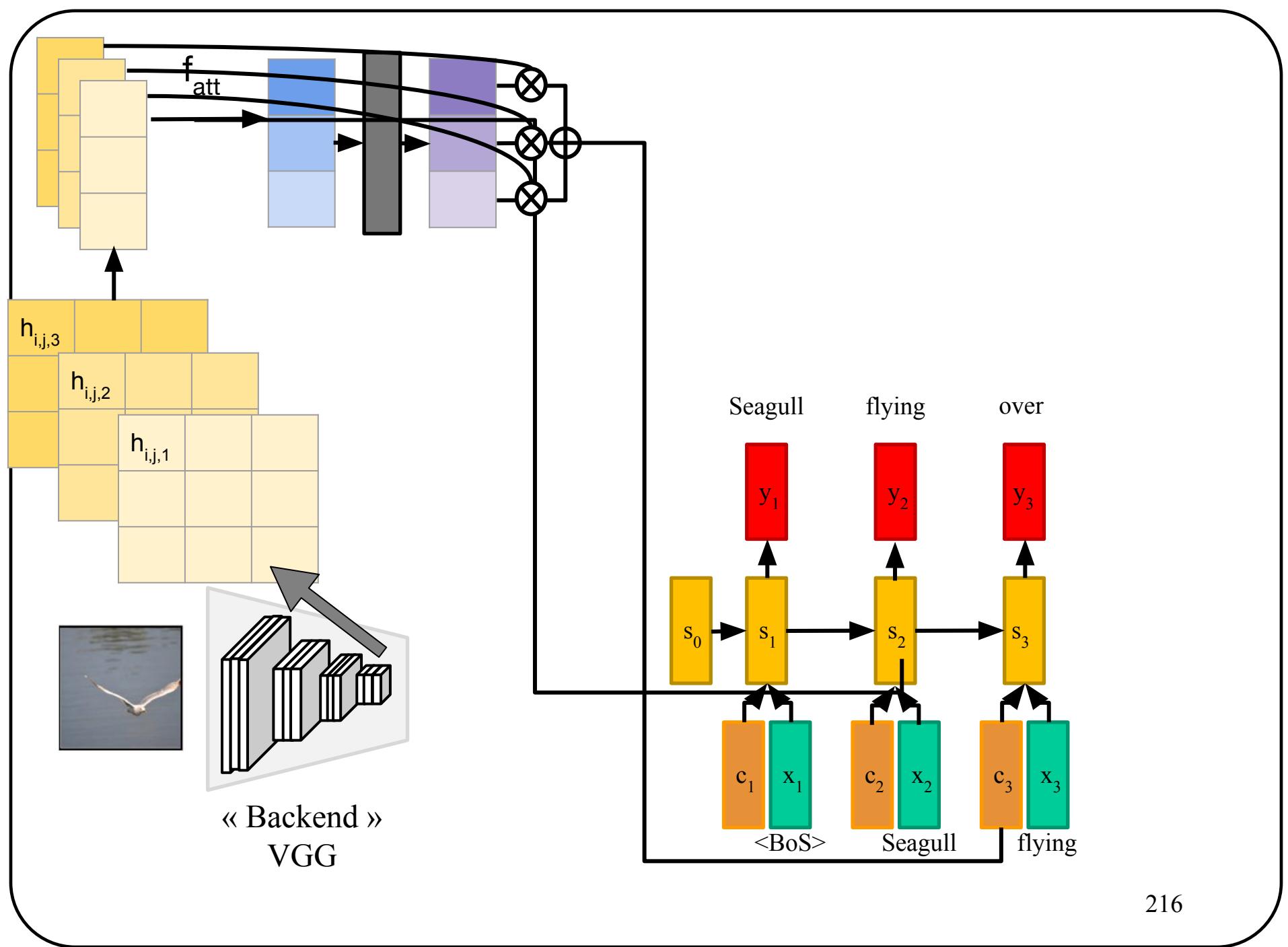


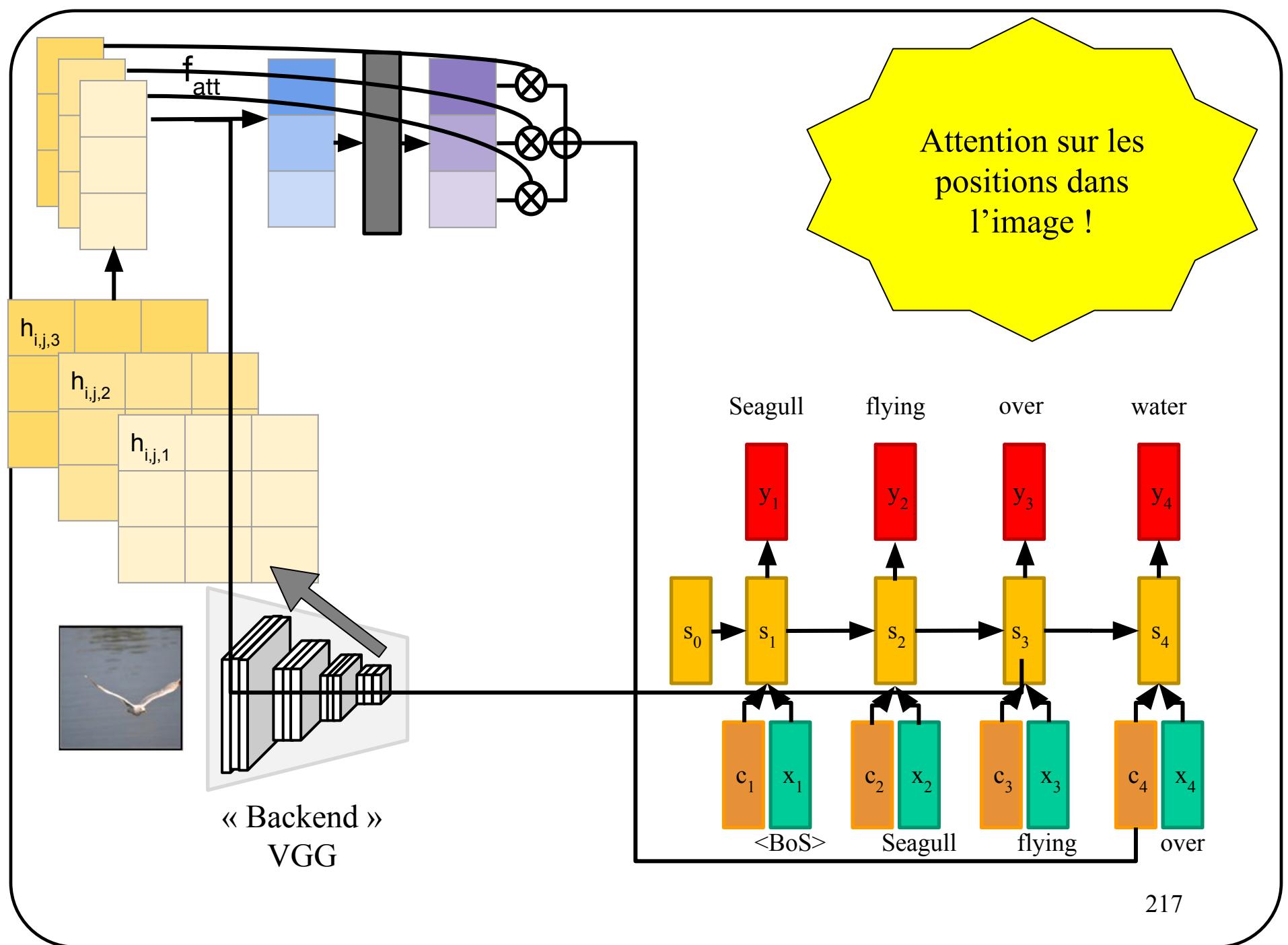


e,a ont la même “forme” que h









Soft attention



Hard attention



A

bird

flying

over

a

body

of

water

.

Soft attention



Hard attention



A

bird

flying

over

a

body

of

water

.

Attention portée initialement sur l'oiseau, puis autour de l'oiseau



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



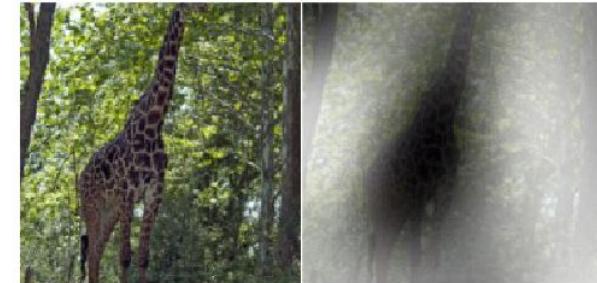
A stop sign is on a road with a mountain in the background.



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

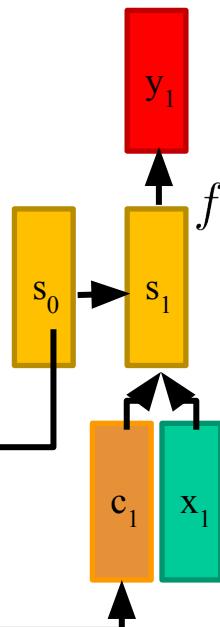
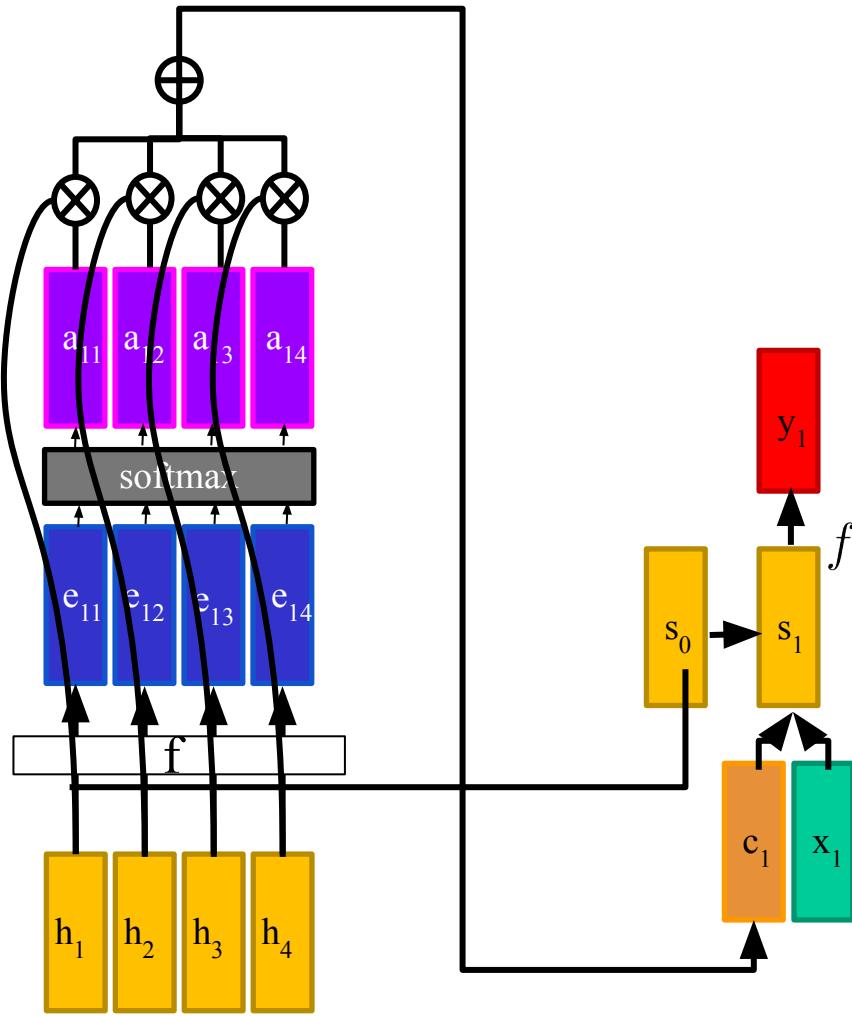


A group of people sitting on a boat in the water.



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

Attention: quelques modifications



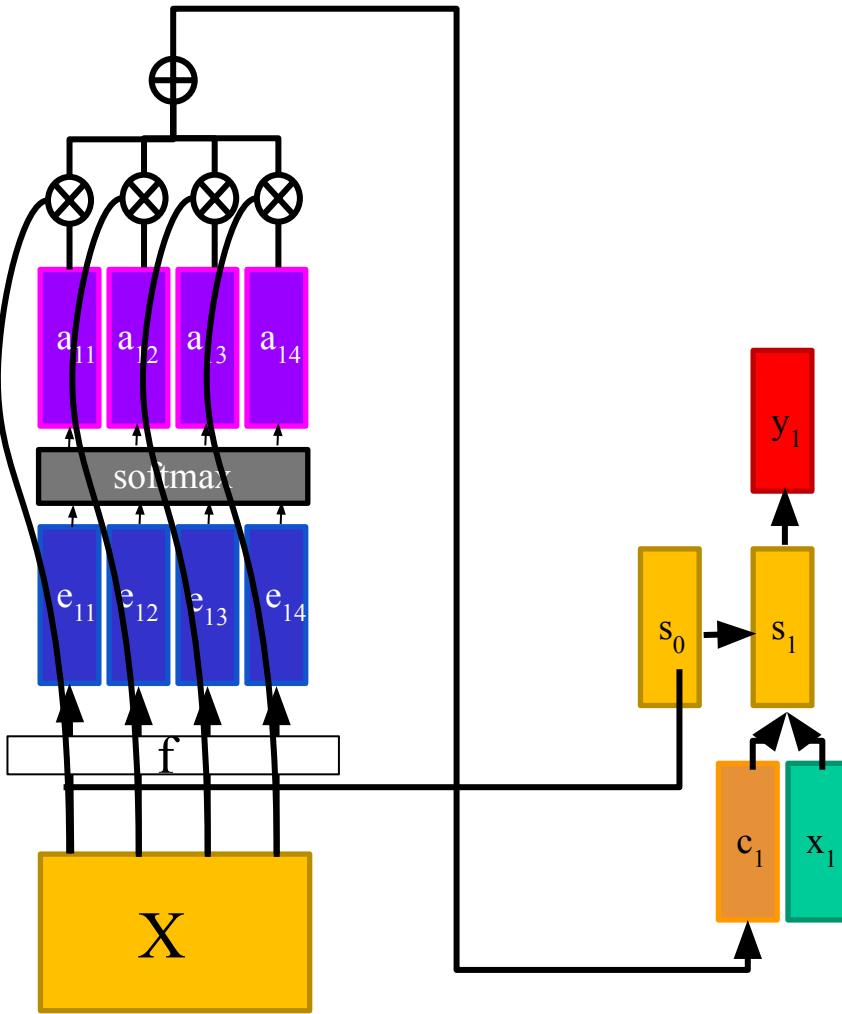
$$e_{ti} = f(s_{t-1}, h_i)$$

$$f(s_{t-1}, h_i) = \vec{v}_a \tanh(W^{sa} \vec{s}_{t-1} + W^{ha} \vec{h}_i)$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_{i=1}^T a_{ti} h_i$$

Attention: quelques modifications



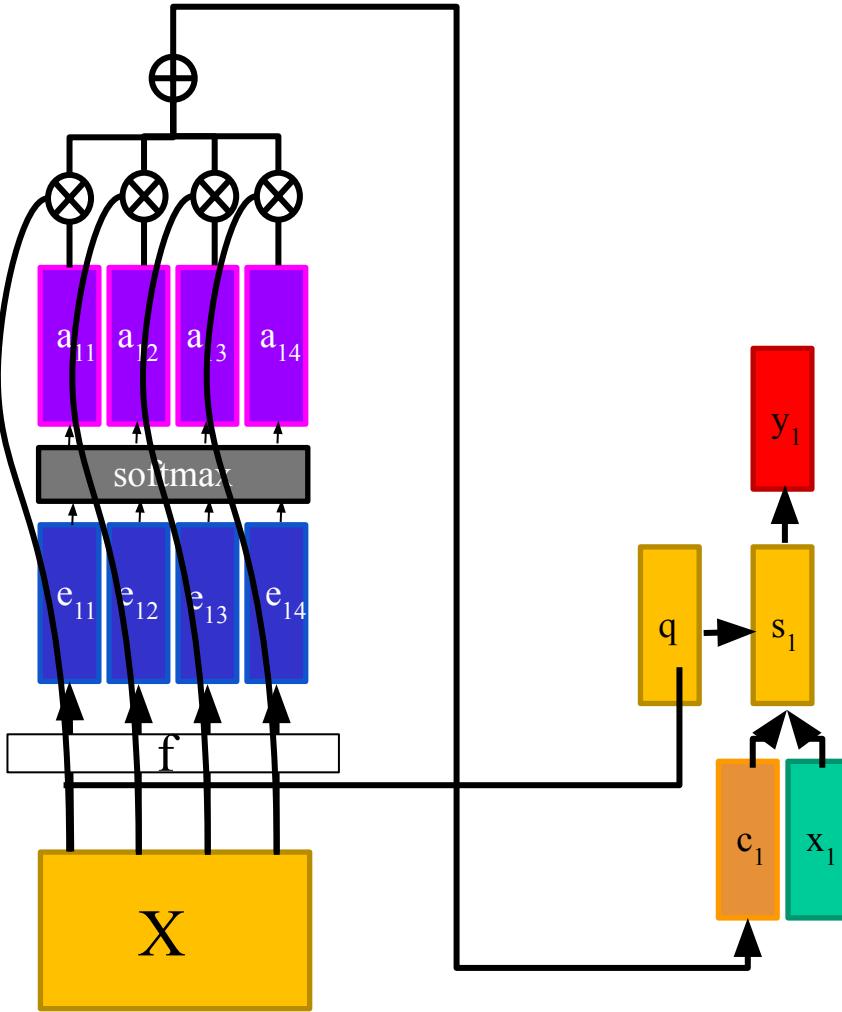
- $h_{1,2,3\dots}$ deviens X

$$e_{ti} = f(s_{t-1}, X_i)$$
$$f(s_{t-1}, X_i) = \vec{v}_a \tanh(W^{qa} \vec{q} + W^{xa} X_i)$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ deviens X
- s_0 deviens q

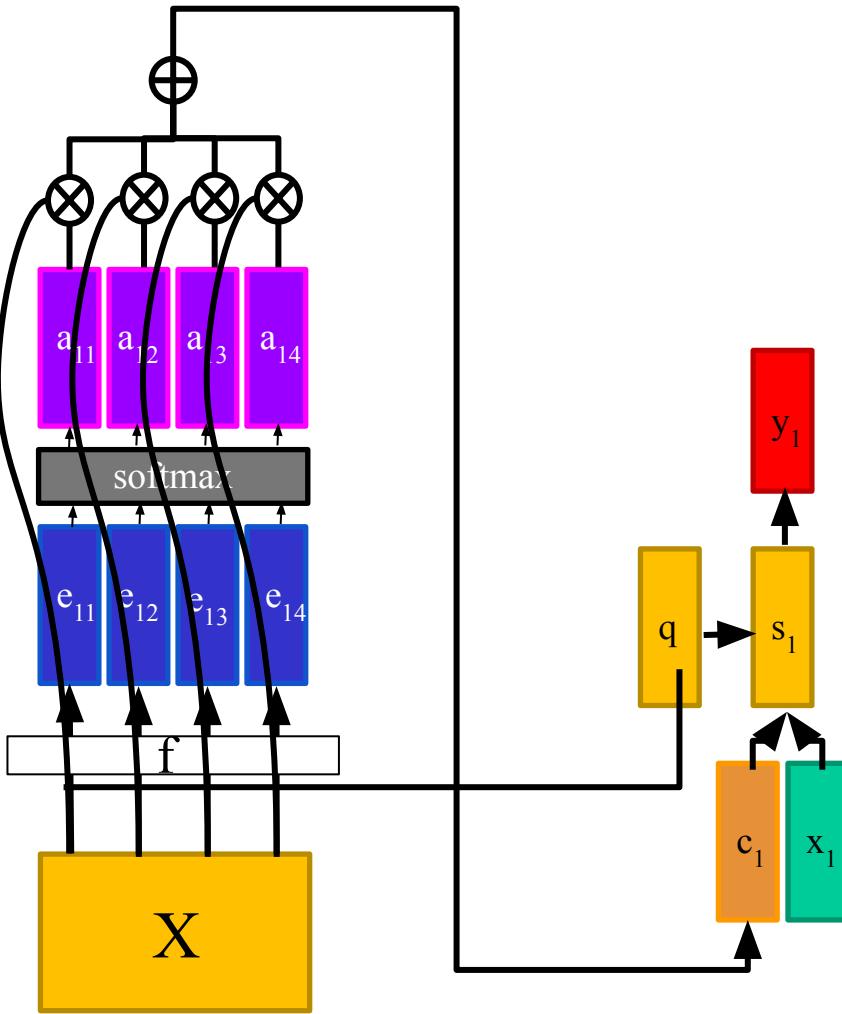
$$e_{ti} = f(q, X_i)$$

$$f(q, X_i) = \vec{v}_a \tanh(W^{qa} \vec{q} + W^{xa} X_i)$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ deviens X
- s_0 deviens q
- f deviens un produit

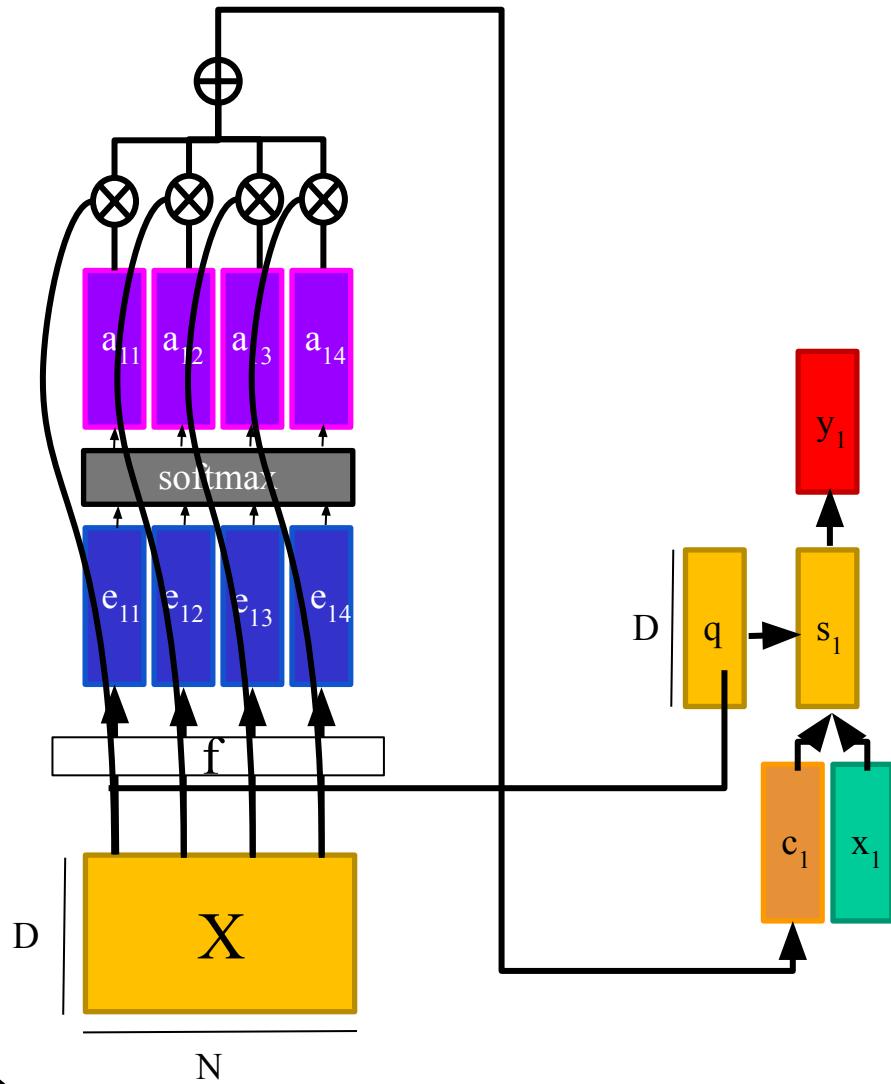
$$e_{ti} = f(q, X_i)$$

$$f(q, X_i) = \vec{q} \cdot X_i$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ deviens X
- s_0 deviens q
- f deviens un produit normalisé

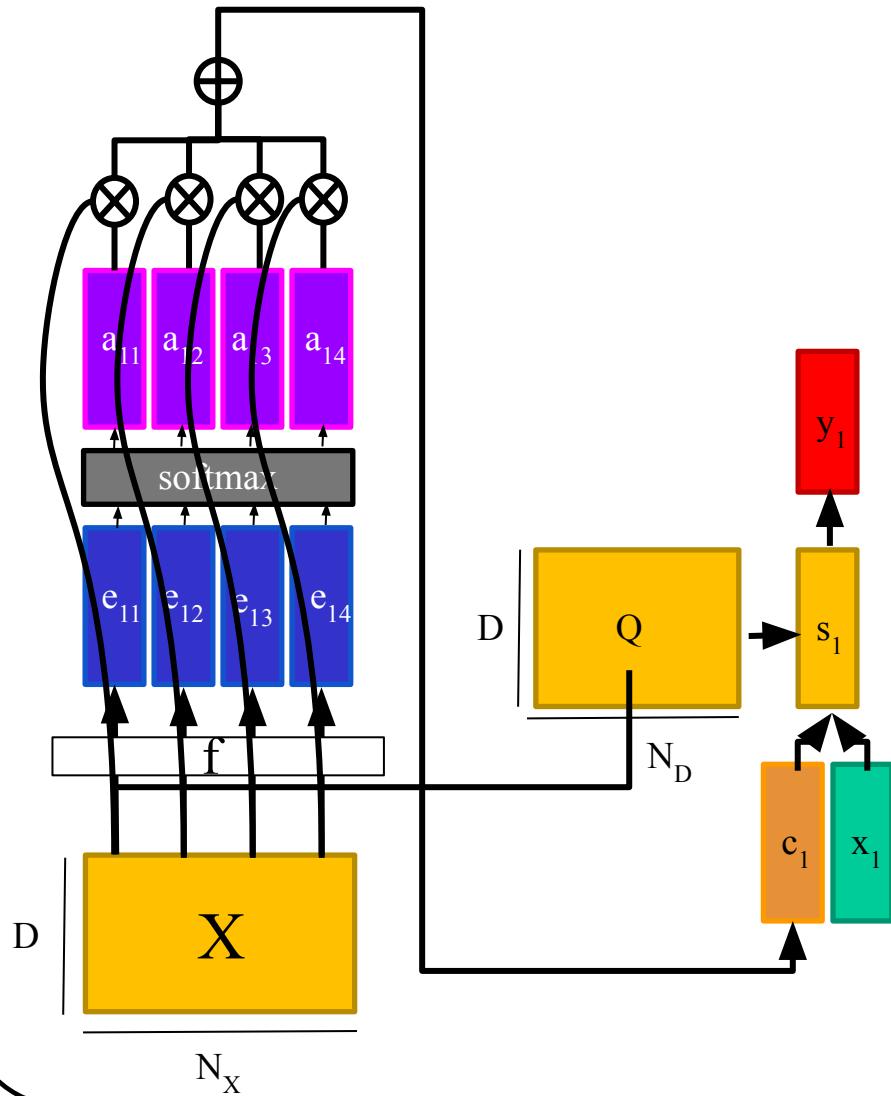
$$e_{ti} = f(q, X_i)$$

$$f(q, X_i) = (\vec{q} \cdot X_i) / \sqrt{D}$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ deviens X
- s_0 deviens multiples $q \rightarrow Q$
- f deviens un produit normalisé

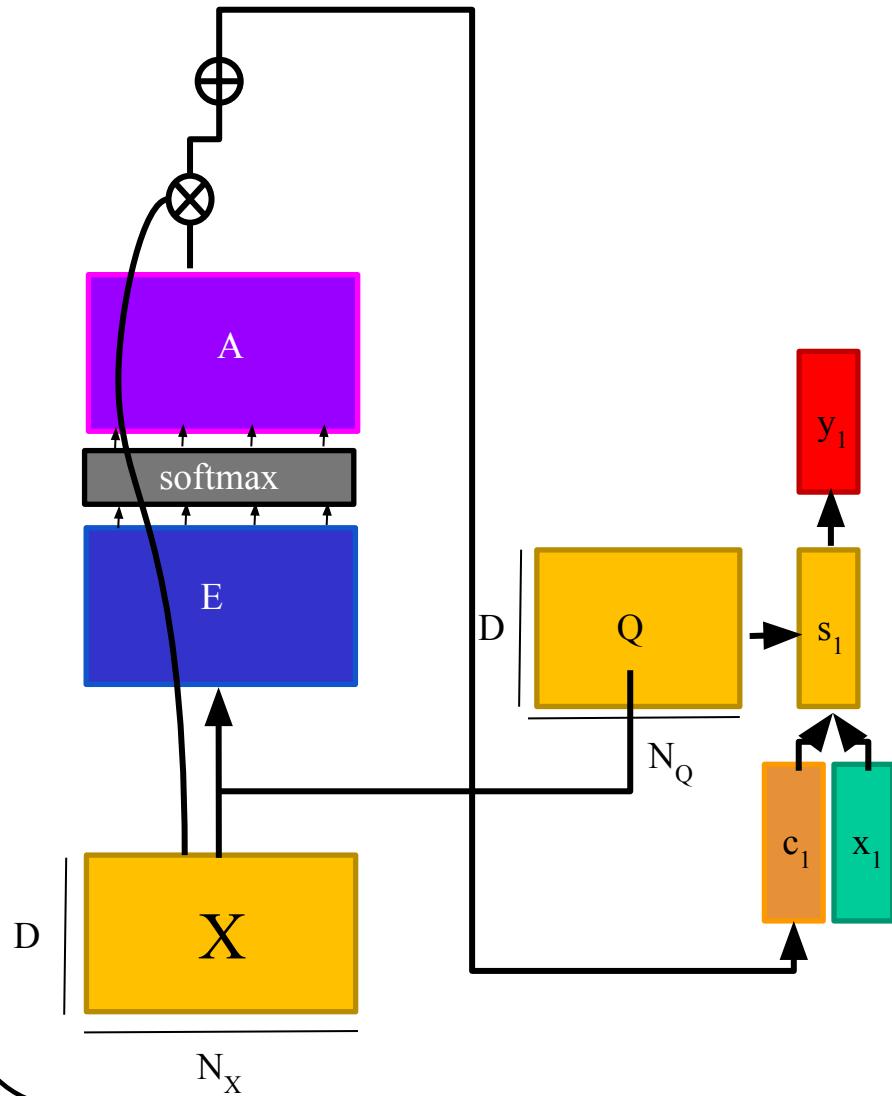
$$e_{ti} = f(q, X_i)$$

$$f(q, X_i) = (\vec{q} \cdot X_i) / \sqrt{D}$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^T \exp(e_{tj})}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



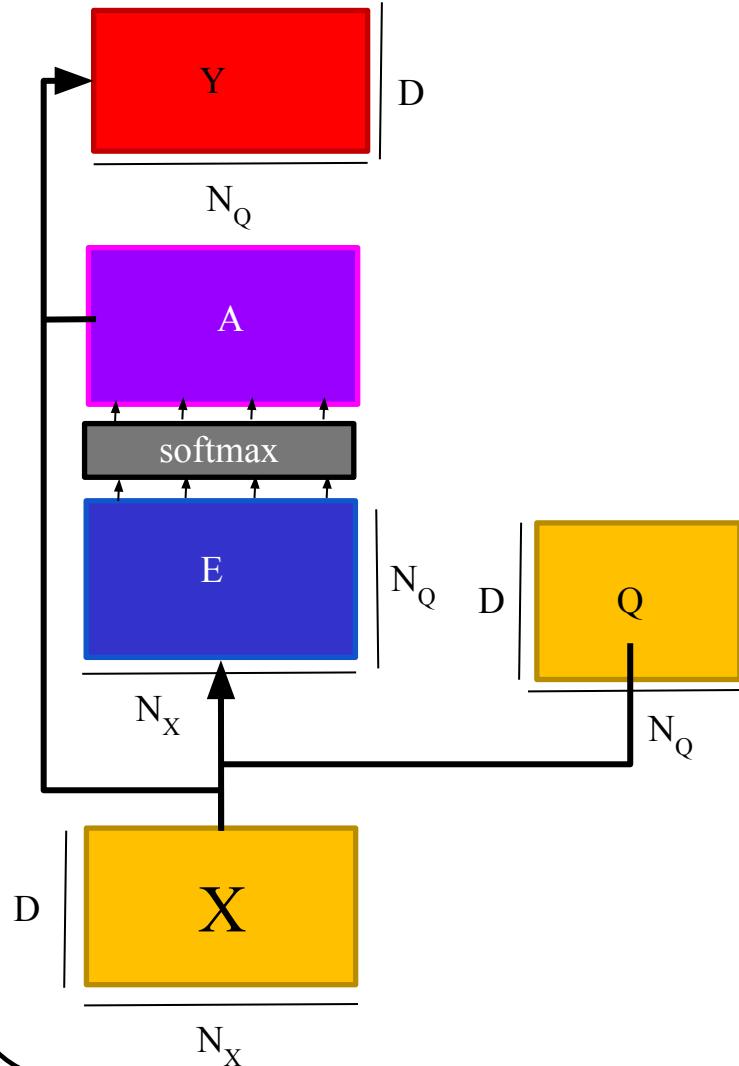
- $h_{1,2,3\dots}$ devient X
- s_0 devient multiples $q \rightarrow Q$
- E est un produit matrice-matrice normalisé

$$E = QX^T / \sqrt(D)$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_x} \exp(e_j)}$$

$$c_t = \sum_i a_{ti} X_i$$

Attention: quelques modifications



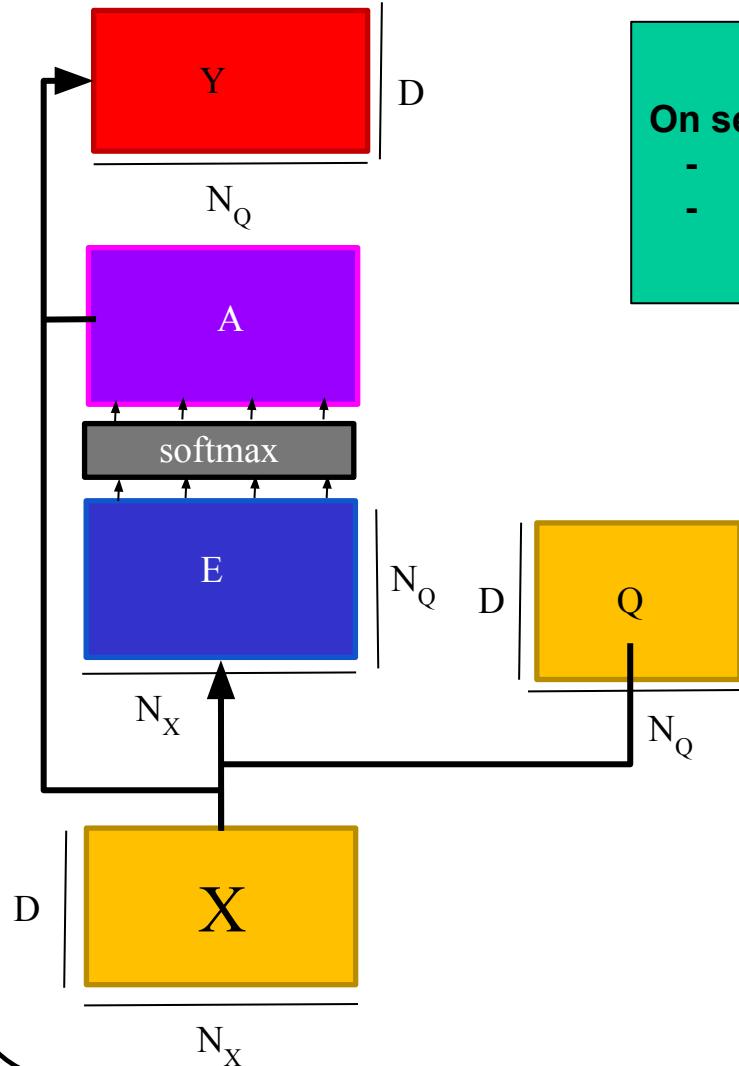
- $h_{1,2,3\dots}$ devient X
- s_0 devient multiples $q \rightarrow Q$
- E est un produit matrice-matrice normalisé
- La sortie est un produit matrice-matrice

$$E = QX^T / \sqrt(D)$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_x} \exp(e_j)}$$

$$Y = AX$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ devient X

On se sert de la même entrée X pour deux choses:

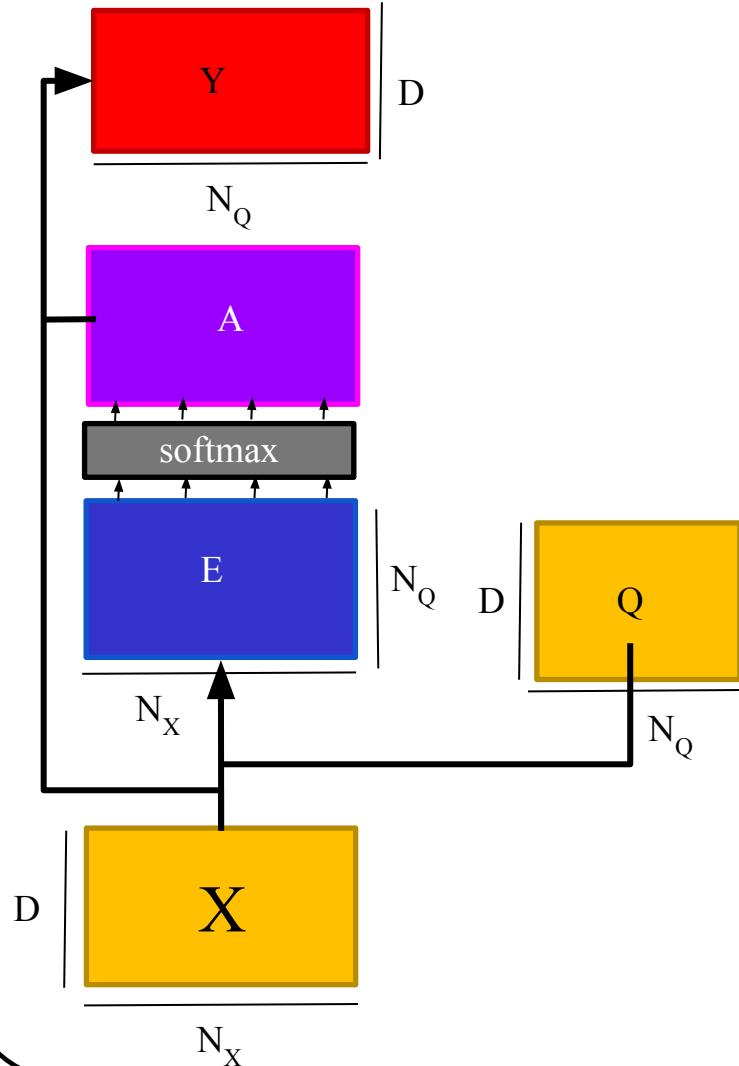
- Calculer l'attention
- Calculer la sortie

On pourrait appliquer
une transformation
différente pour
chaque fonction !

$$Y = AX$$

$$\sqrt{D} \cdot \text{xp}(e_j)$$

Attention: quelques modifications



- $h_{1,2,3\dots}$ devient X
- s_0 devient multiples $q \rightarrow Q$
- E est un produit matrice-matrice normalisé
- La sortie est un produit matrice-matrice

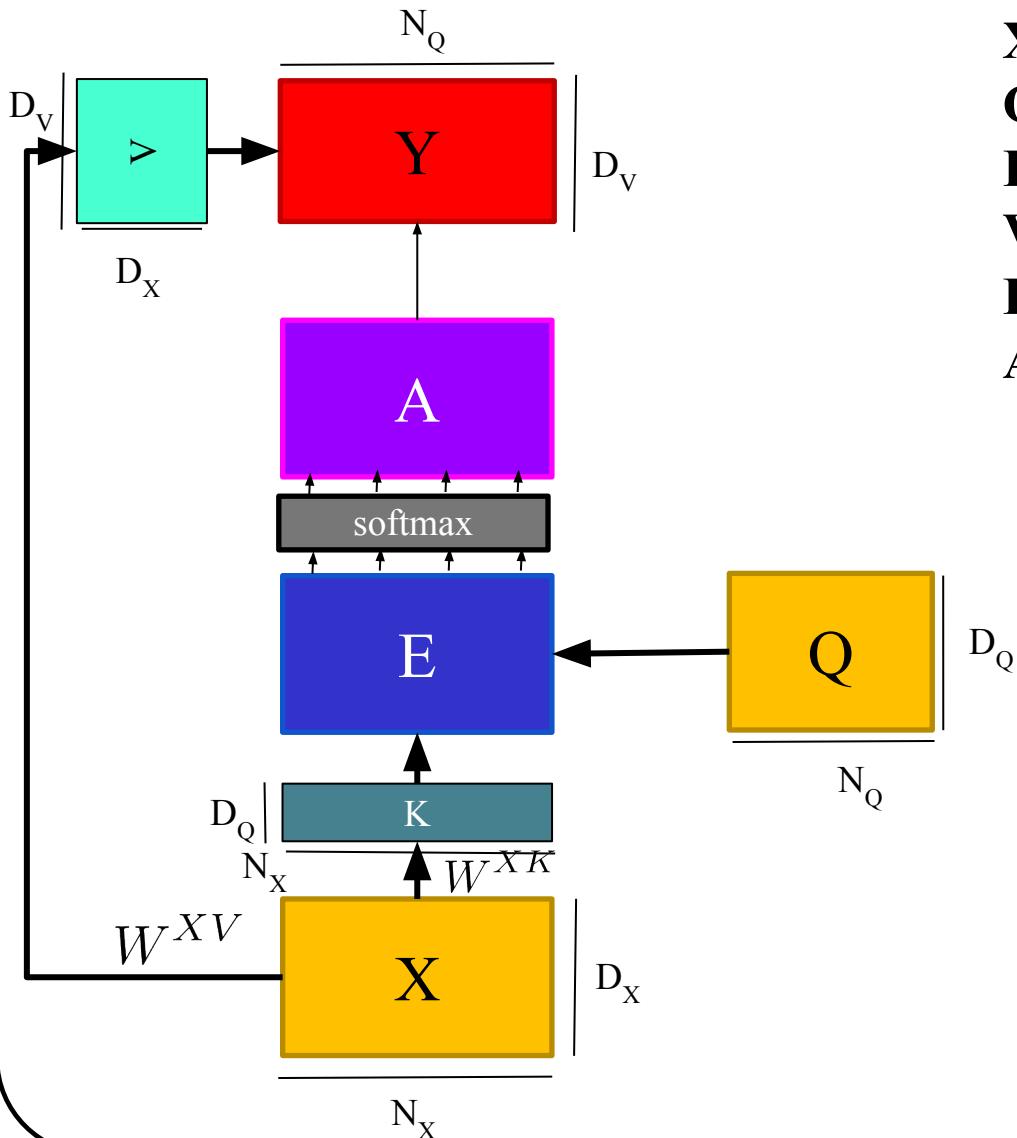
$$E = QX^T / \sqrt(D)$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_x} \exp(e_j)}$$

$$Y = AX$$

Attention

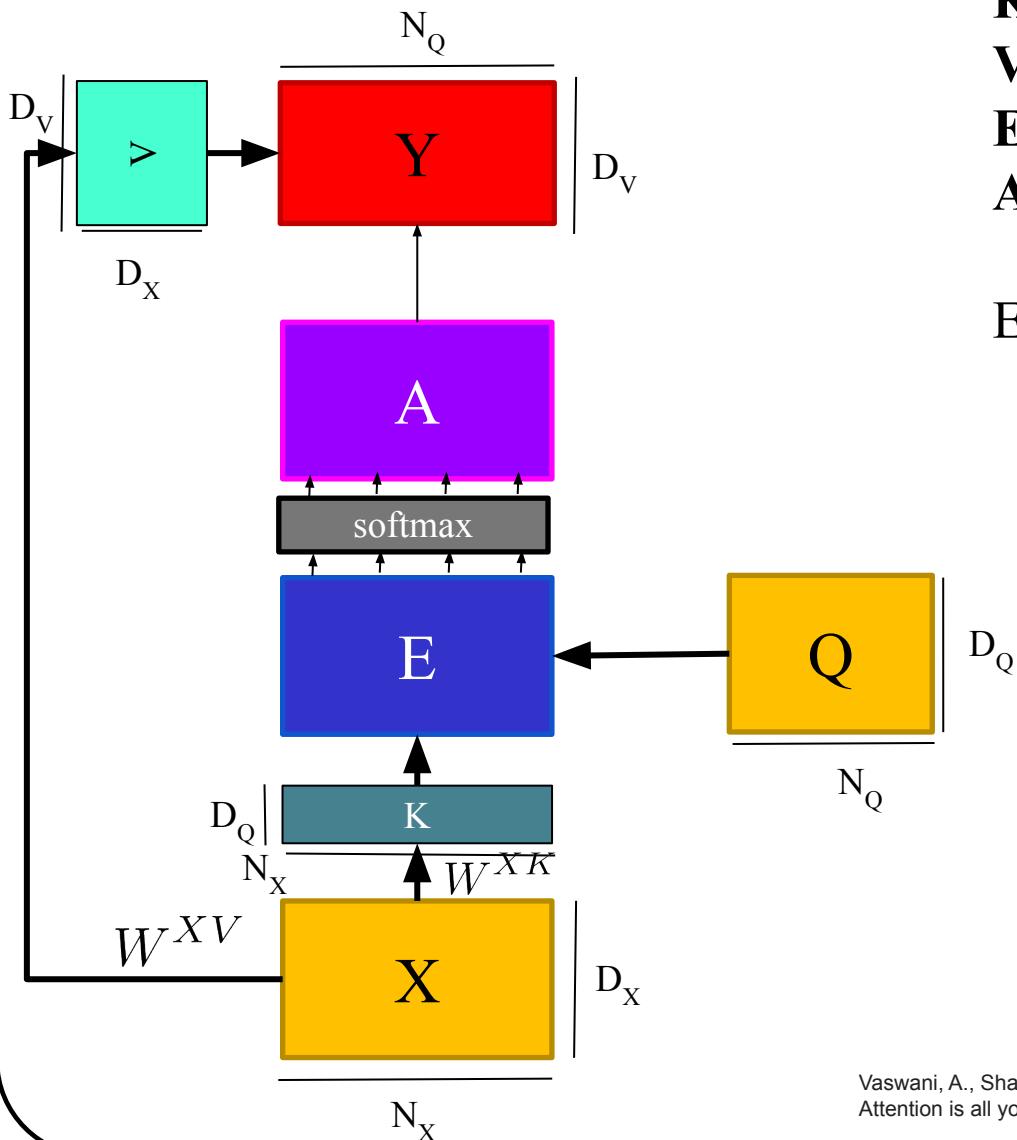
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



X données
Query
Key
Value
Entries ?
Attention

$$K = XW^{XK}$$
$$V = XW^{XV}$$
$$E = QK^T / \sqrt{D}$$
$$A = \frac{\exp(e_i)}{\sum_j^{N_x} \exp(e_j)}$$
$$Y = AV$$

Attention



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

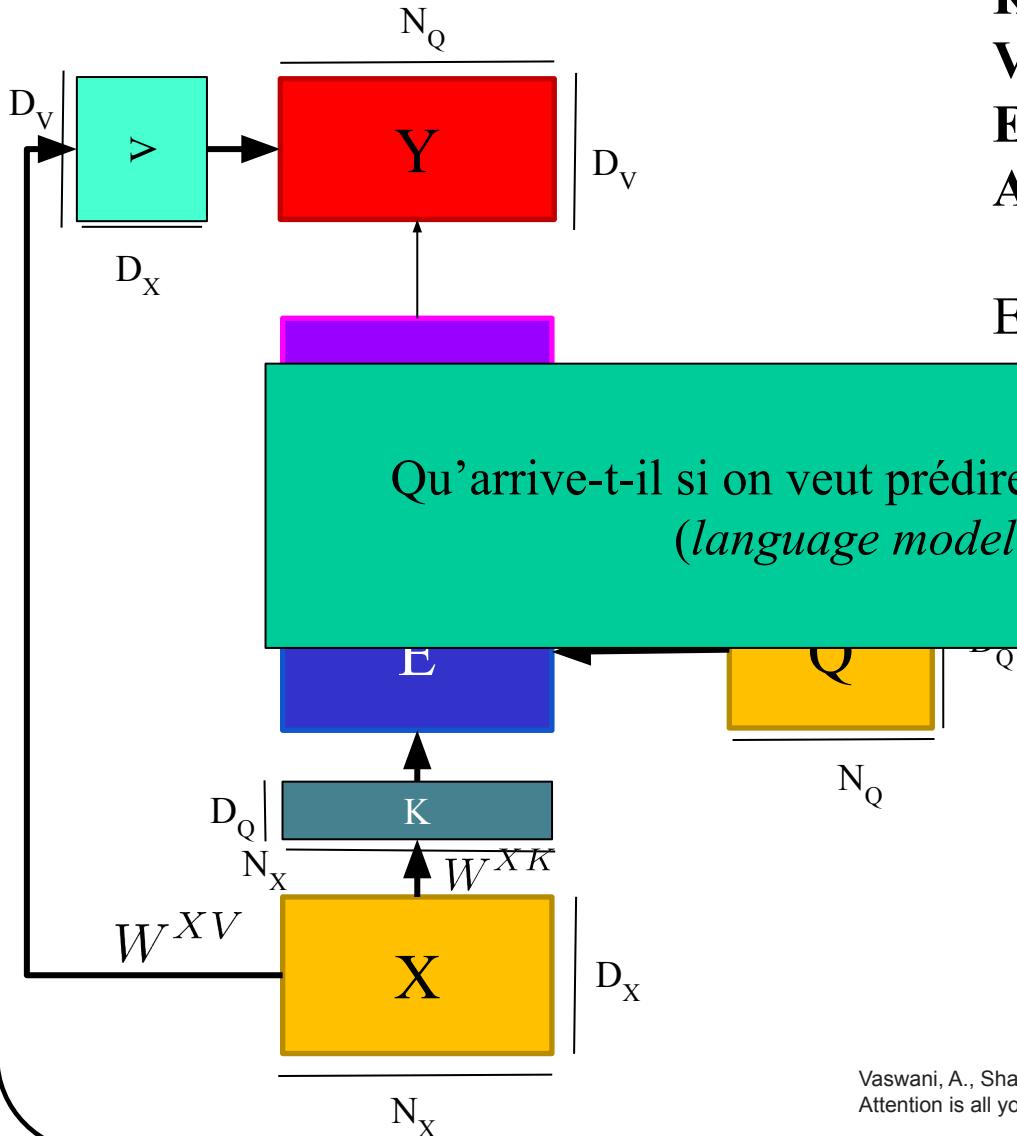
Ex. **X:** phrase en français

Q: <BoS> + phrase en anglais

Y: Phrase en anglais

Pour accommoder la taille fixe, on remplit la fin par un caractère spécial “<pad>”

Attention



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

Ex. **X:** phrase en français

en anglais

Qu’arrive-t-il si on veut prédire le prochain mot ?

(*language modelling*)

Si la taille fixe, on

caractère

spécial “<pad>”

Self-attention (intra-attention)

X données

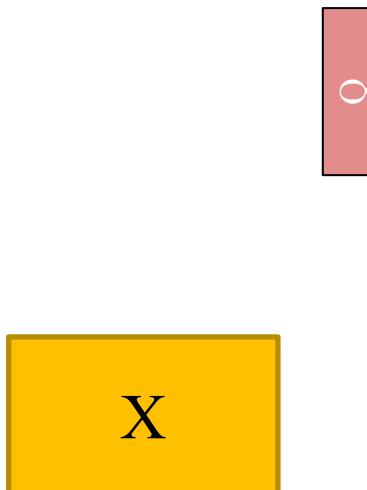
Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées



Self-attention (intra-attention)

X données

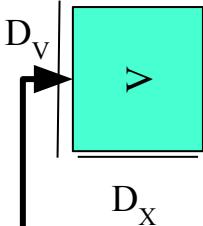
Query: que chercher dans nos données

Key: la “clé” des données

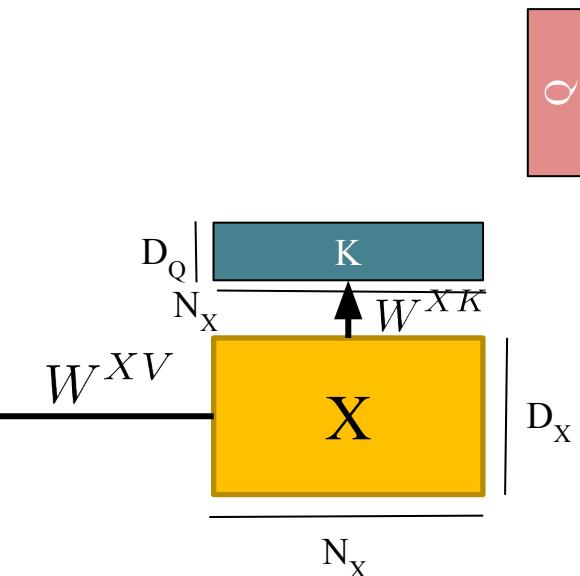
Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées



$$K = XW^{XK}$$
$$V = XW^{XV}$$



Self-attention (intra-attention)

X données

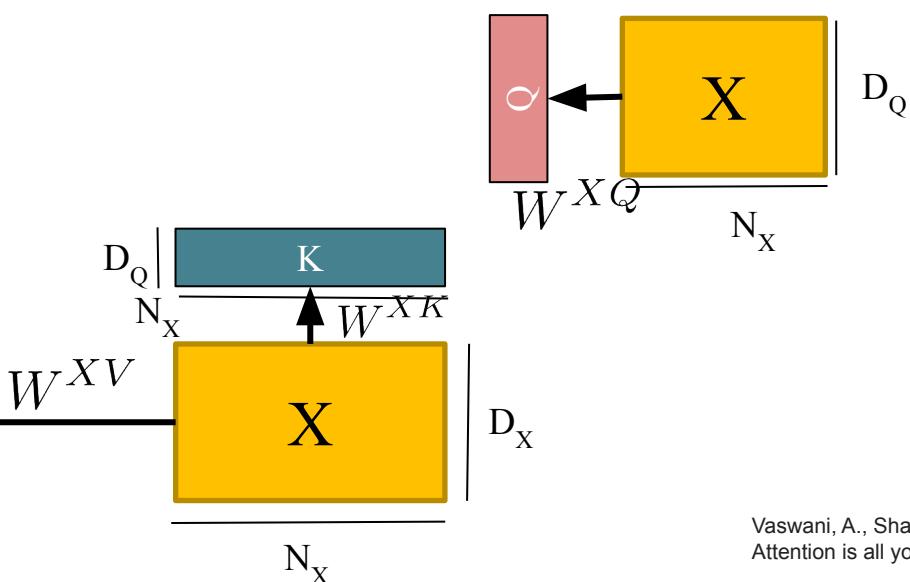
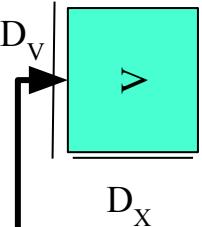
Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées



$$K = XW^{XK}$$
$$V = XW^{XV}$$
$$Q = XW^{XQ}$$

Self-attention (intra-attention)

X données

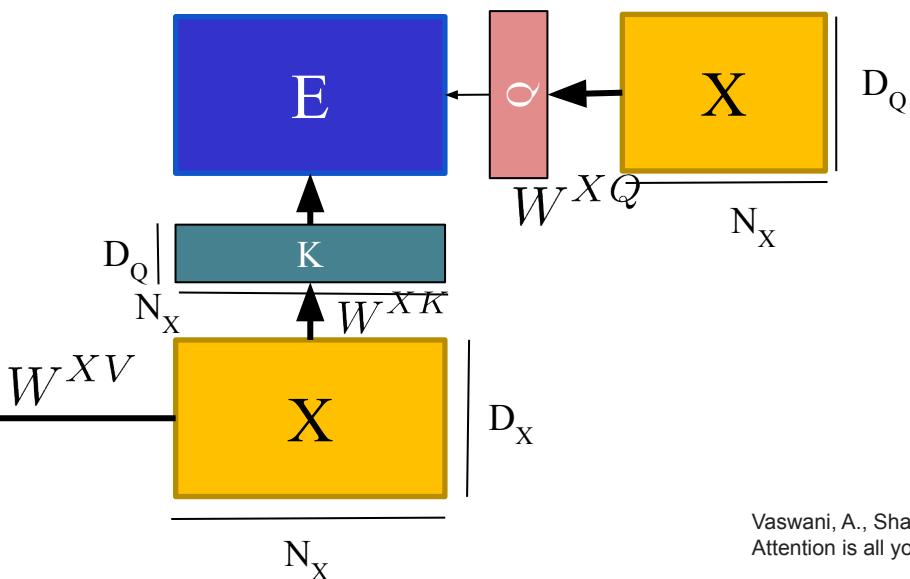
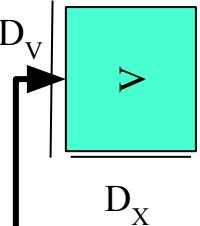
Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées



$$\begin{aligned} K &= XW^{XK} \\ V &= XW^{XV} \\ Q &= XW^{XQ} \\ E &= QK^T / \sqrt(D) \end{aligned}$$

Self-attention (intra-attention)

X données

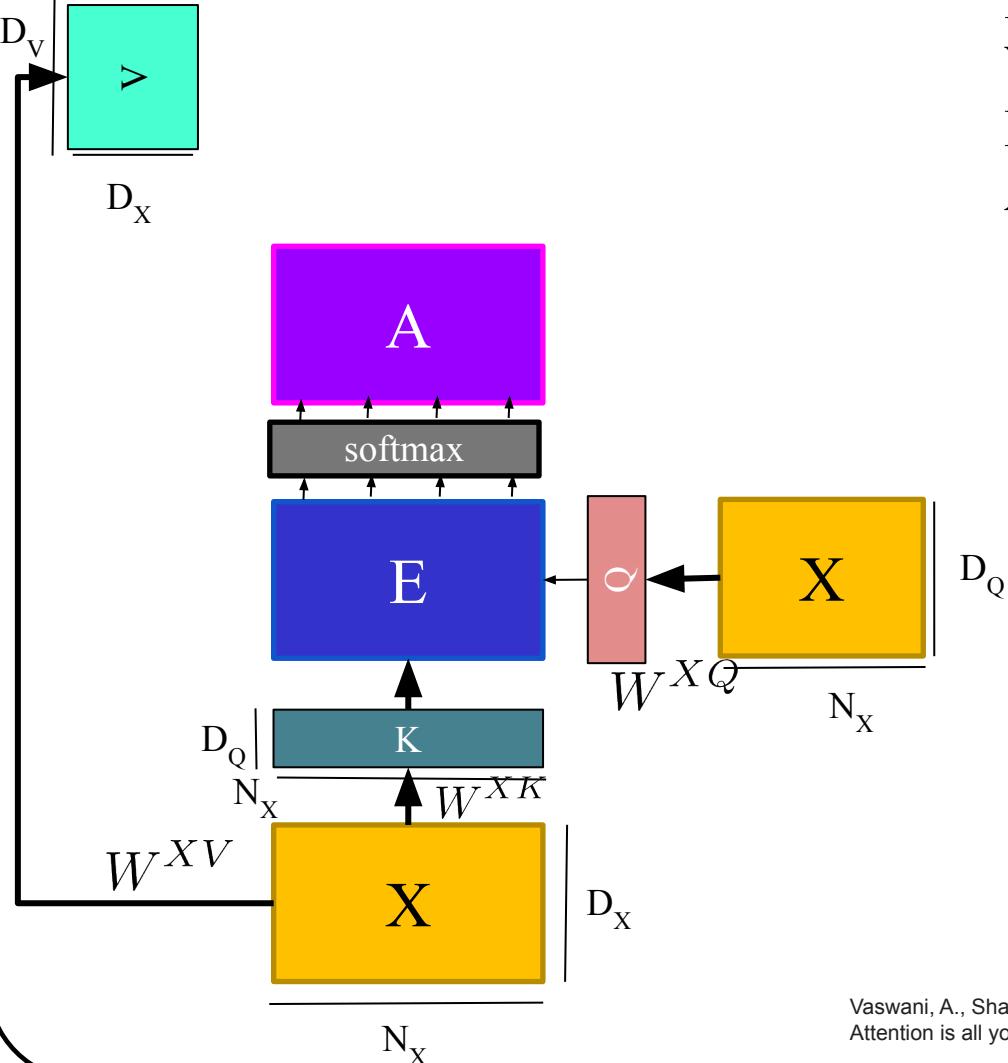
Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

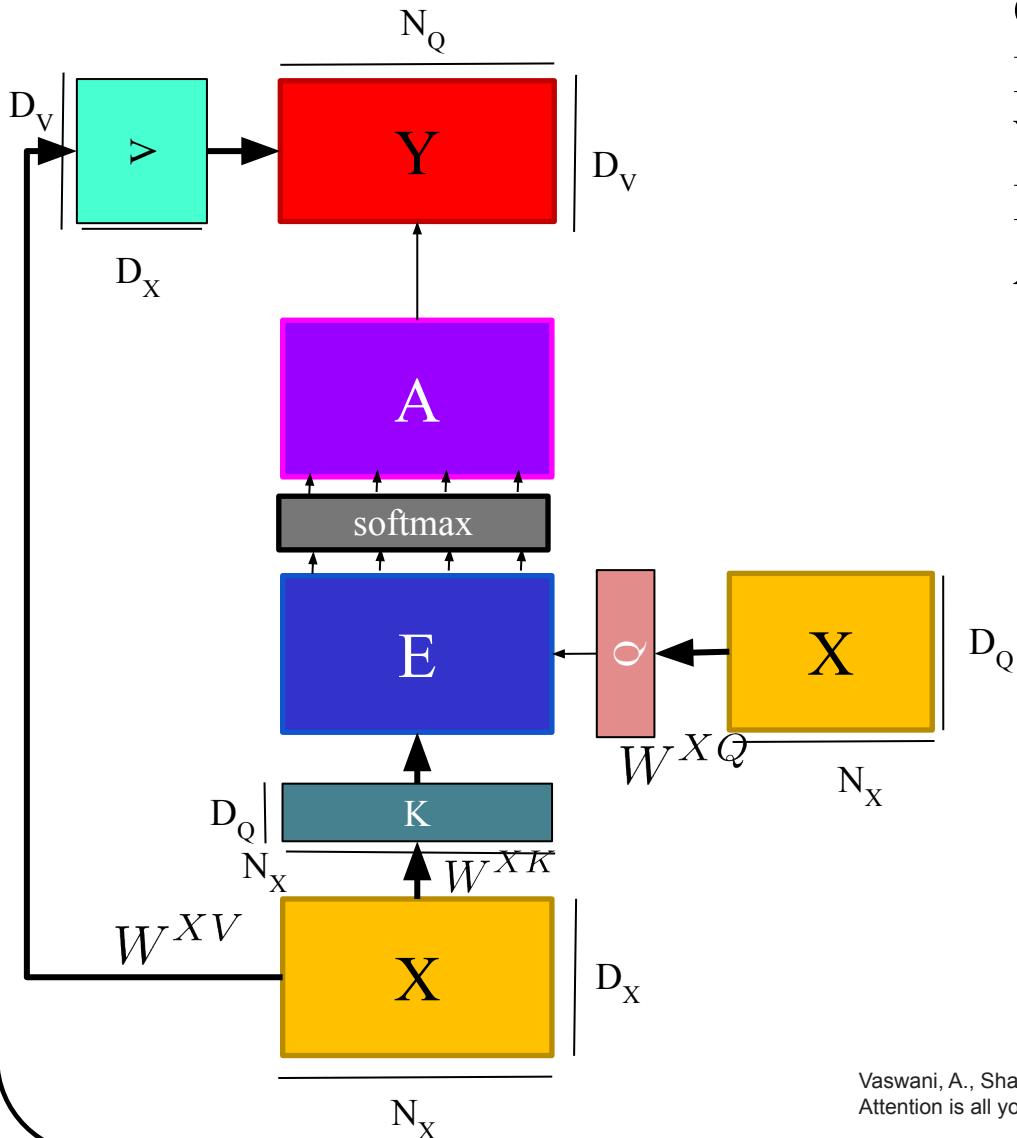
Entries: le “contenu” des données

Attention: la pertinence des entrées



$$K = XW^{XK}$$
$$V = XW^{XV}$$
$$Q = XW^{XQ}$$
$$E = QK^T / \sqrt(D)$$
$$A = \frac{\exp(e_i)}{\sum_j^{N_X} \exp(e_j)}$$

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

$$K = XW^{XK}$$

$$V = XW^{XV}$$

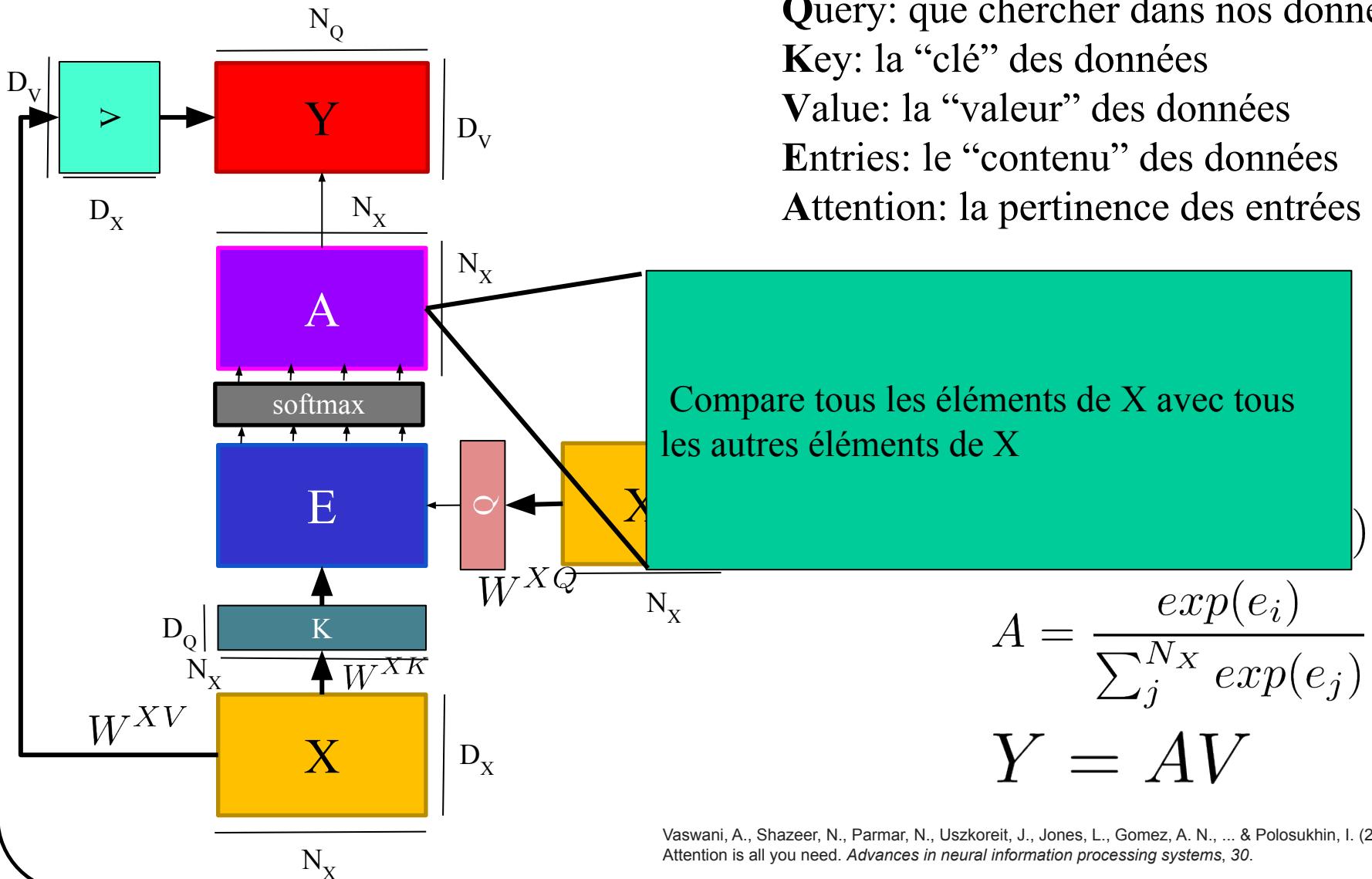
$$Q = XW^{XQ}$$

$$E = QK^T / \sqrt{D}$$

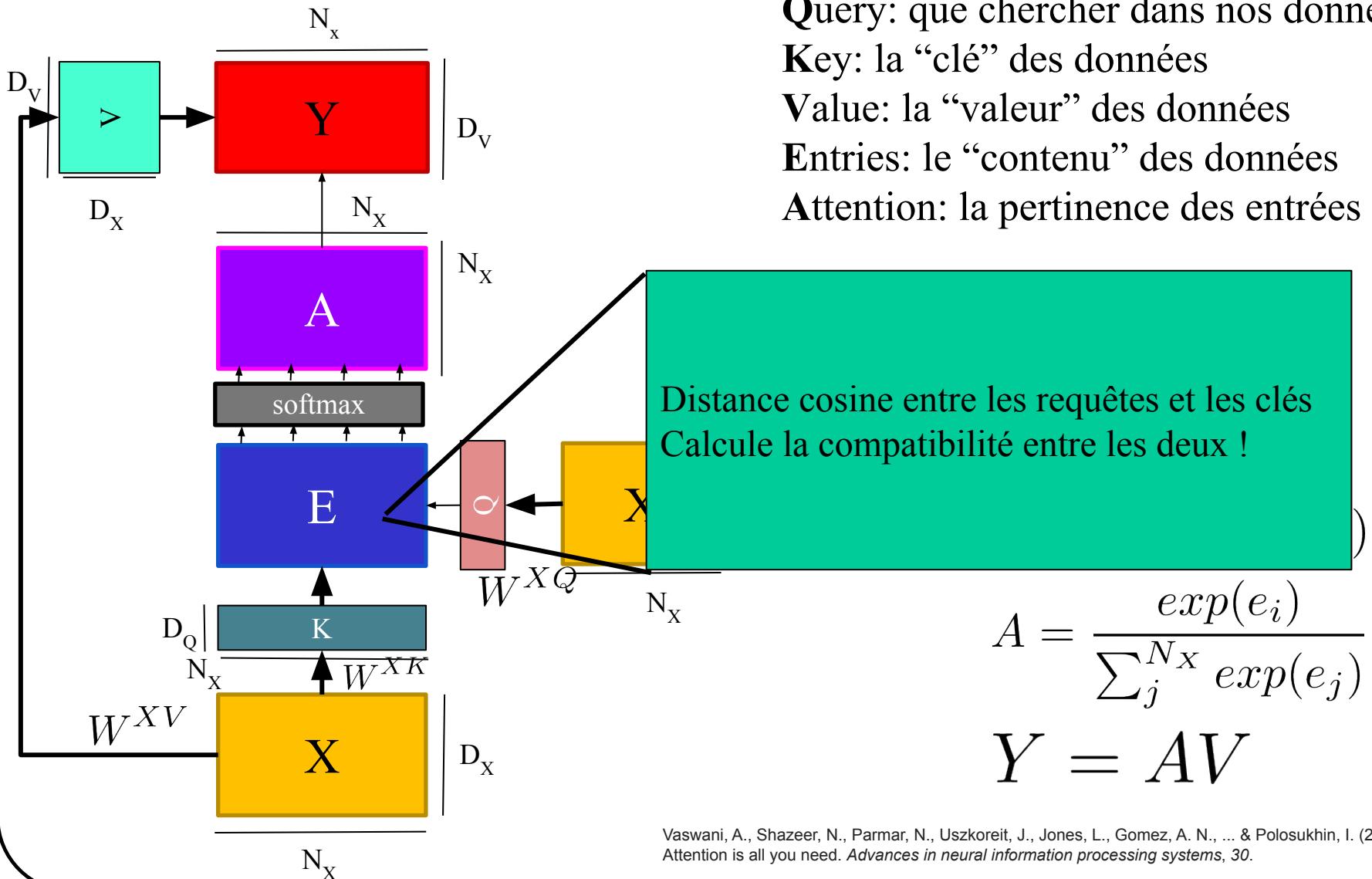
$$A = \frac{\exp(e_i)}{\sum_j^{N_X} \exp(e_j)}$$

$$Y = AV$$

Self-attention (intra-attention)

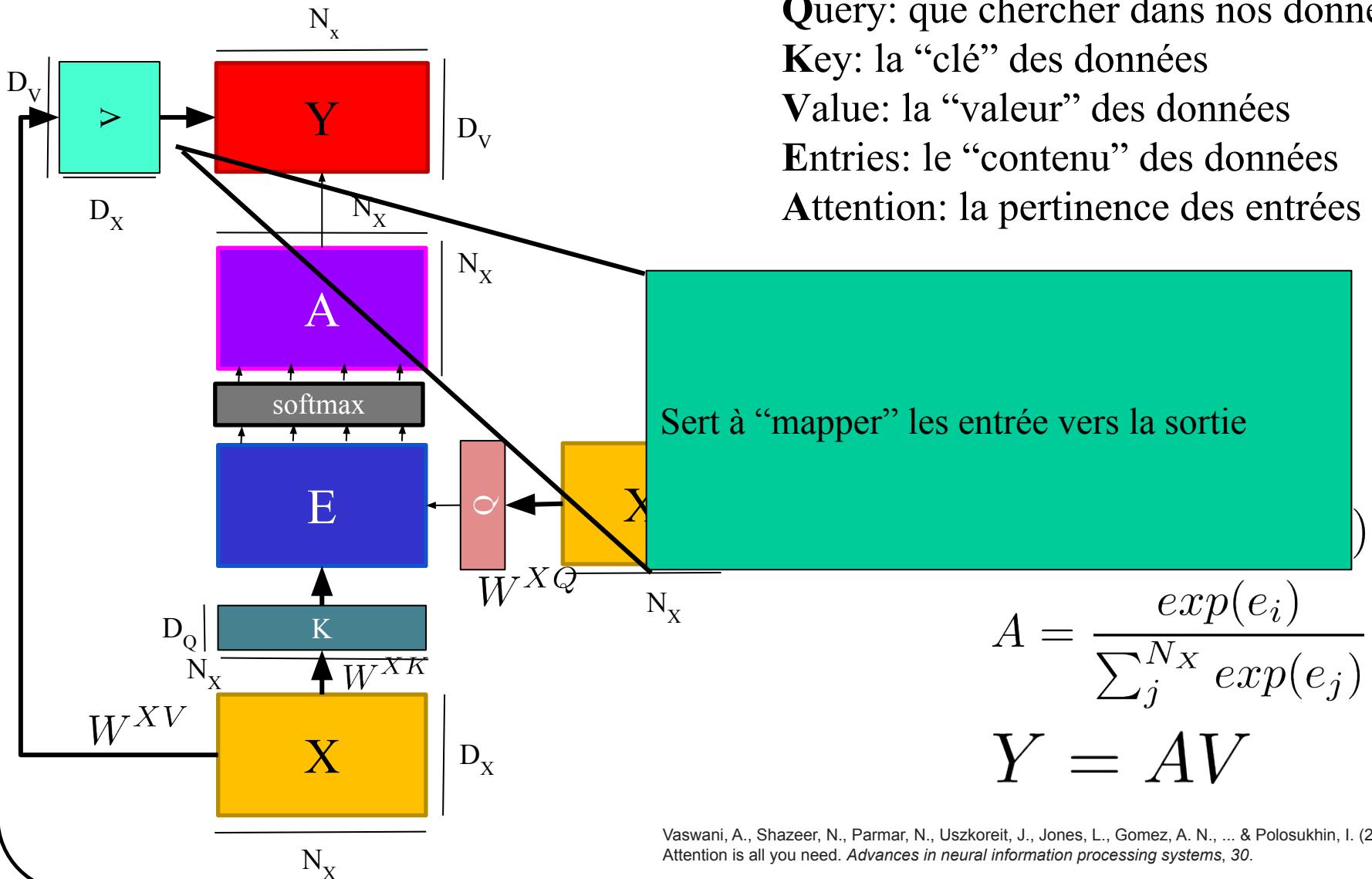


Self-attention (intra-attention)



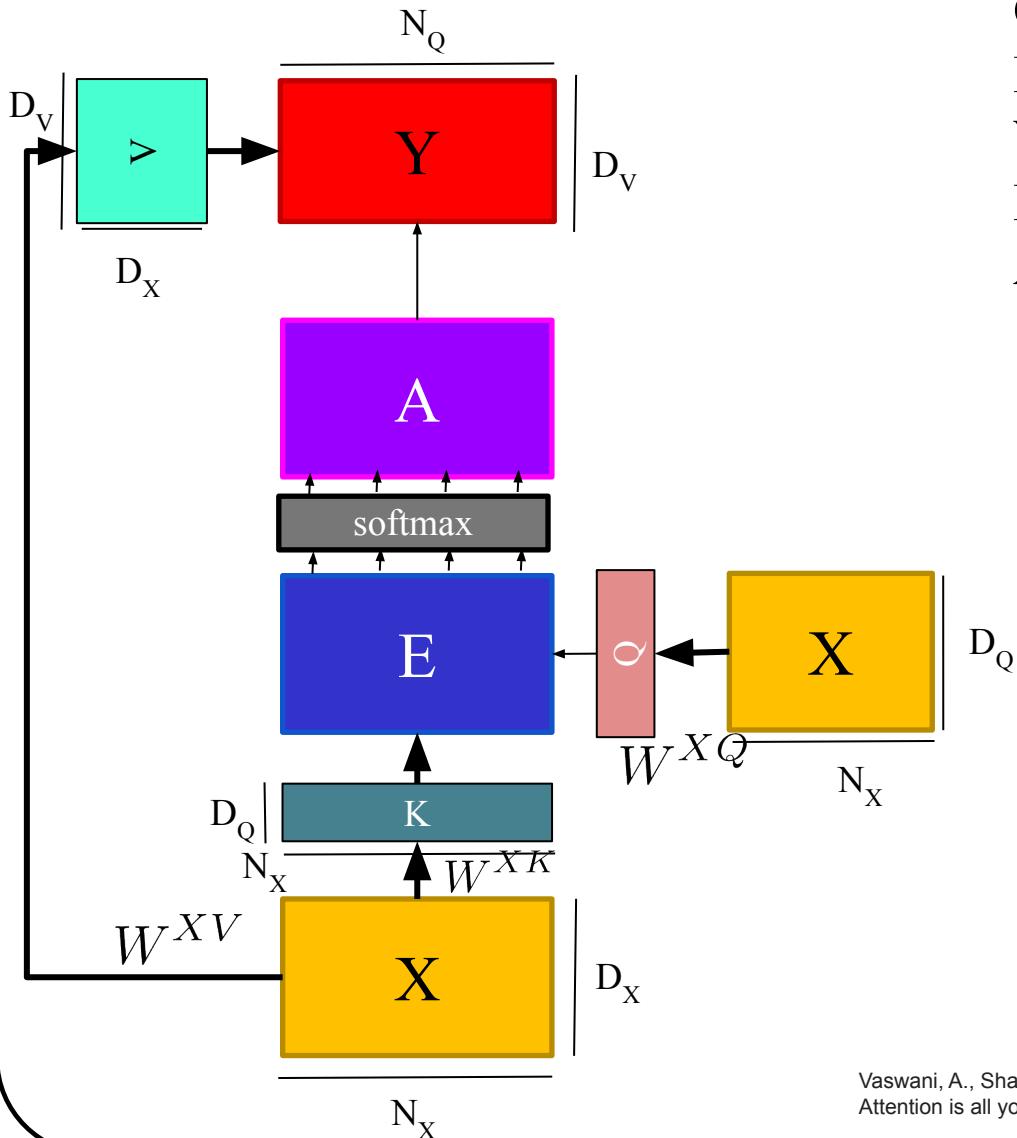
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Self-attention (intra-attention)



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017).
Attention is all you need. *Advances in neural information processing systems*, 30.

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

$$K = XW^{XK}$$

$$V = XW^{XV}$$

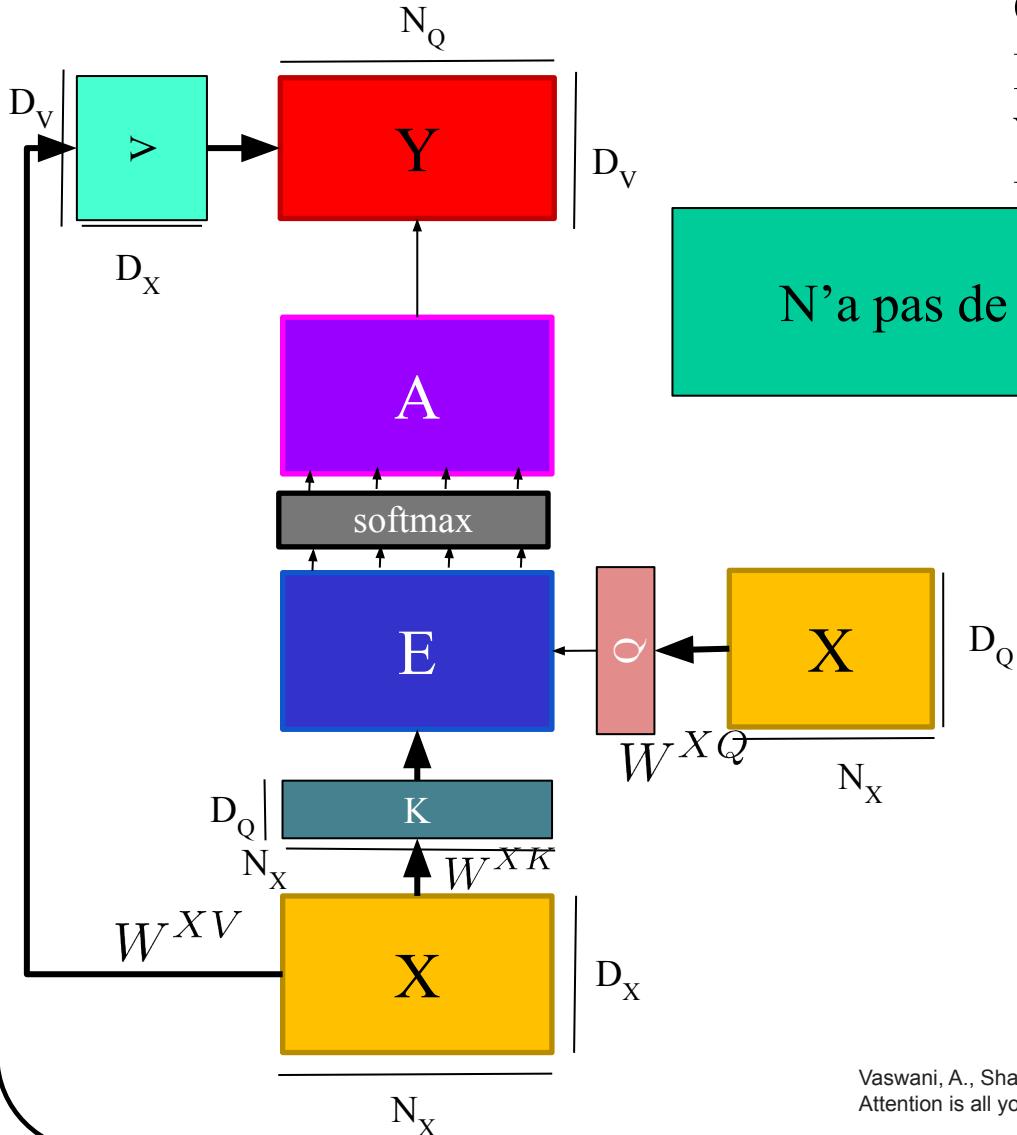
$$Q = XW^{XQ}$$

$$E = QK^T / \sqrt{D}$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_X} \exp(e_j)}$$

$$Y = AV$$

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données
ce des entrées

N'a pas de notion d'ordre !

$$K = XW^{XK}$$

$$V = XW^{XV}$$

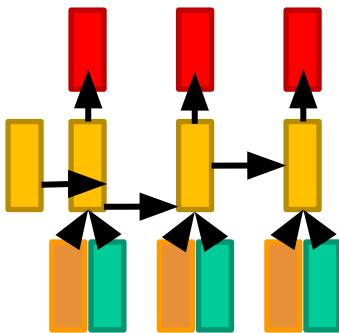
$$Q = XW^{XQ}$$

$$E = QK^T / \sqrt{D}$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_x} \exp(e_j)}$$

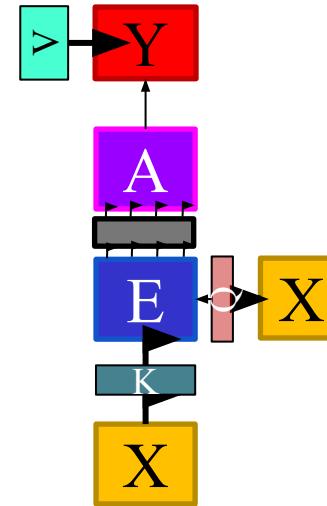
$$Y = AV$$

Self-attention (intra-attention)



“Il dit qu’il y a un problème”

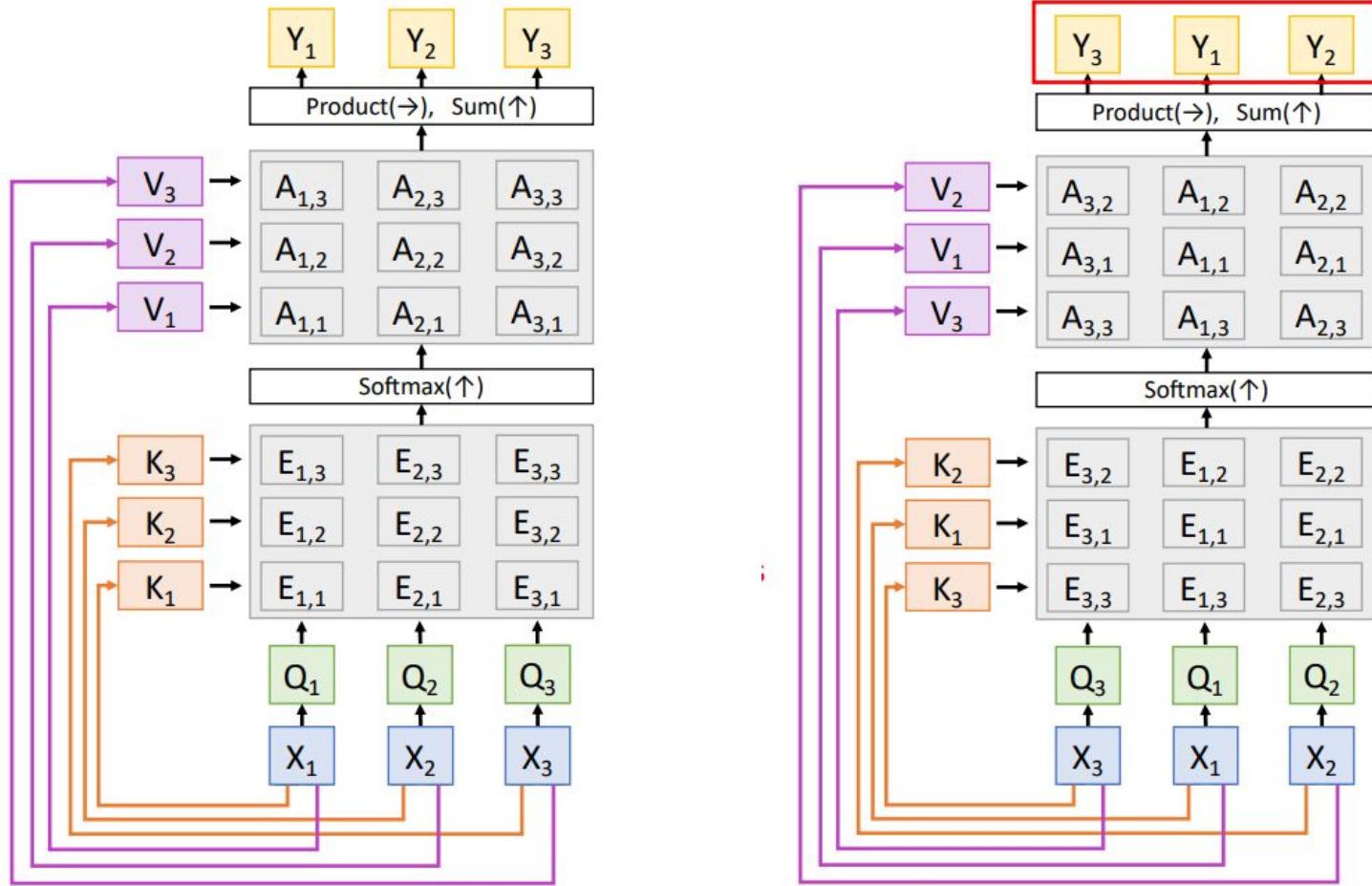
Le vecteur “ h ” encode différemment les “il” grâce à la récurrent



“Il dit qu’il y a un problème”

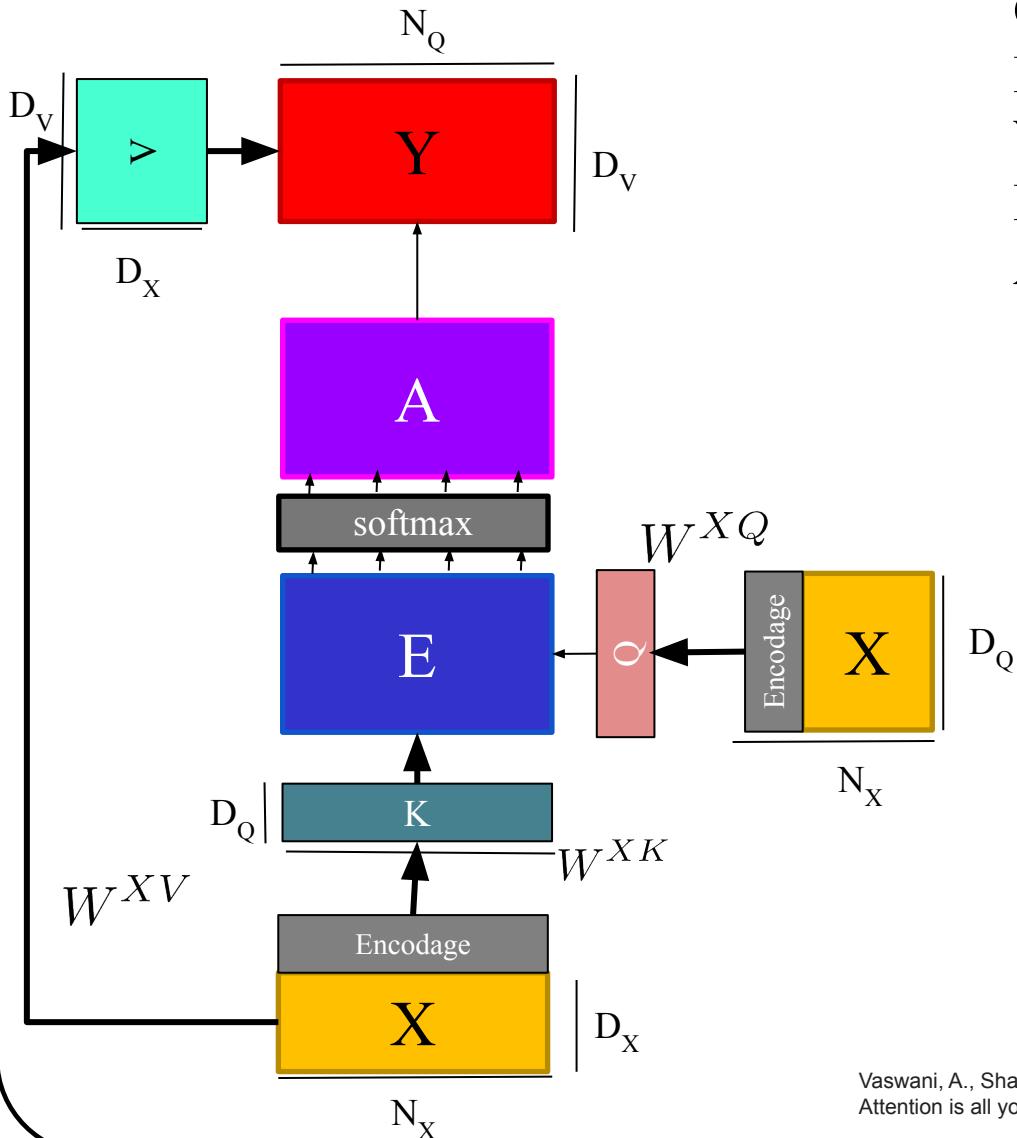
Les deux “il” ont la même représentation

Self-attention (intra-attention)



crédit: Justin Johnson

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

$$K = XW^{XK}$$

$$V = XW^{XV}$$

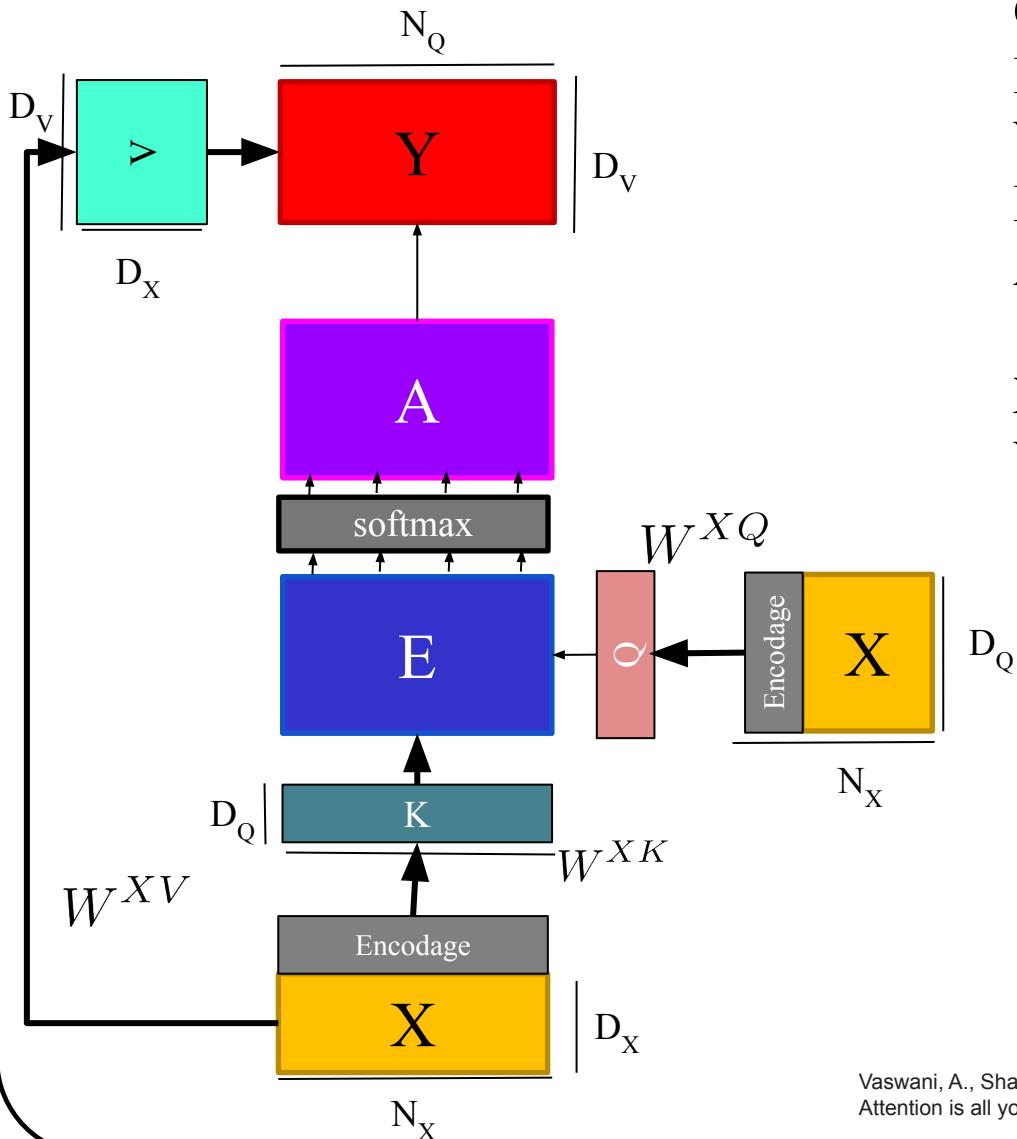
$$Q = XW^{XQ}$$

$$E = QK^T / \sqrt{D}$$

$$A = \frac{\exp(e_i)}{\sum_j^{N_X} \exp(e_j)}$$

$$Y = AV$$

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

Value: la “valeur” des données

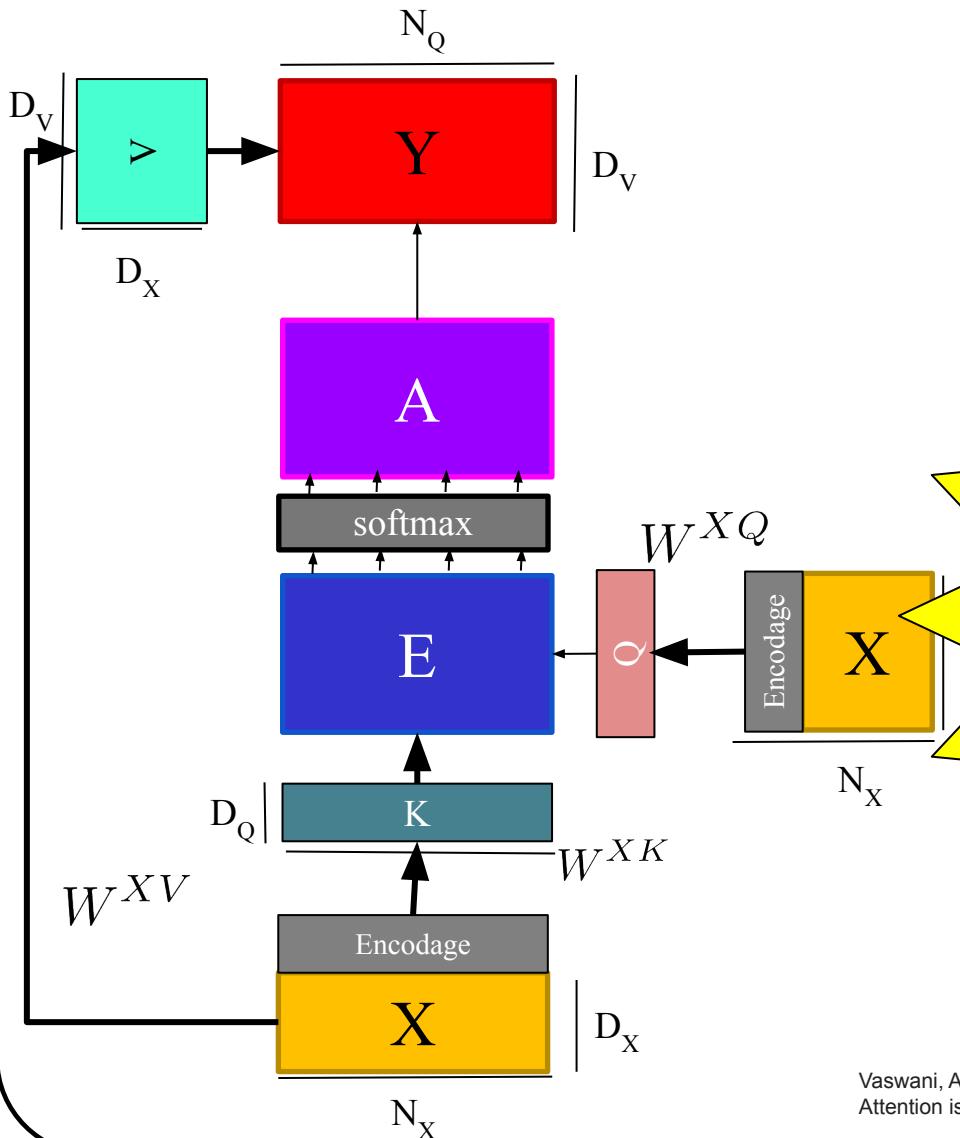
Entries: le “contenu” des données

Attention: la pertinence des entrées

X: <BoS> + Phrase + <pad>

Y: Phrase + <pad>

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

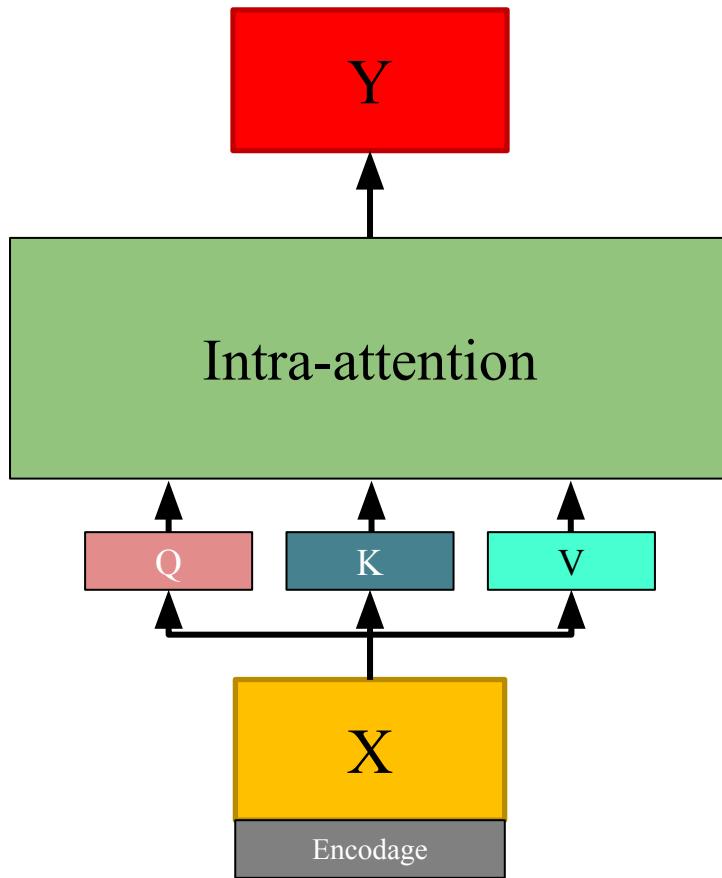
Value: la “valeur” des données

Entries: le “contenu” des données

Attention: la pertinence des entrées

**Mécanisme très général !
Ne fait qu'un mapping vecteur → vecteur**

Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

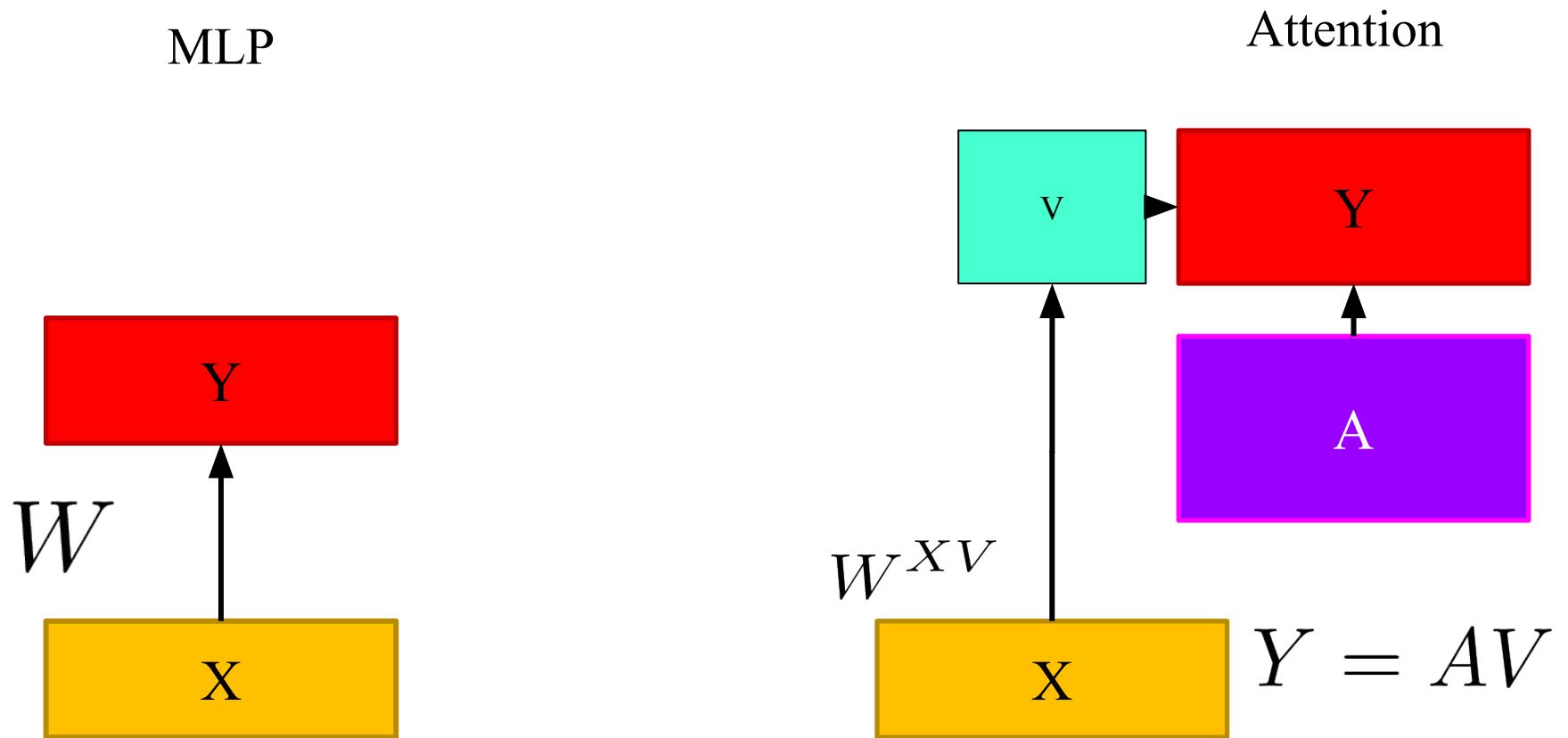
Value: la “valeur” des données

Entries: le “contenu” des données

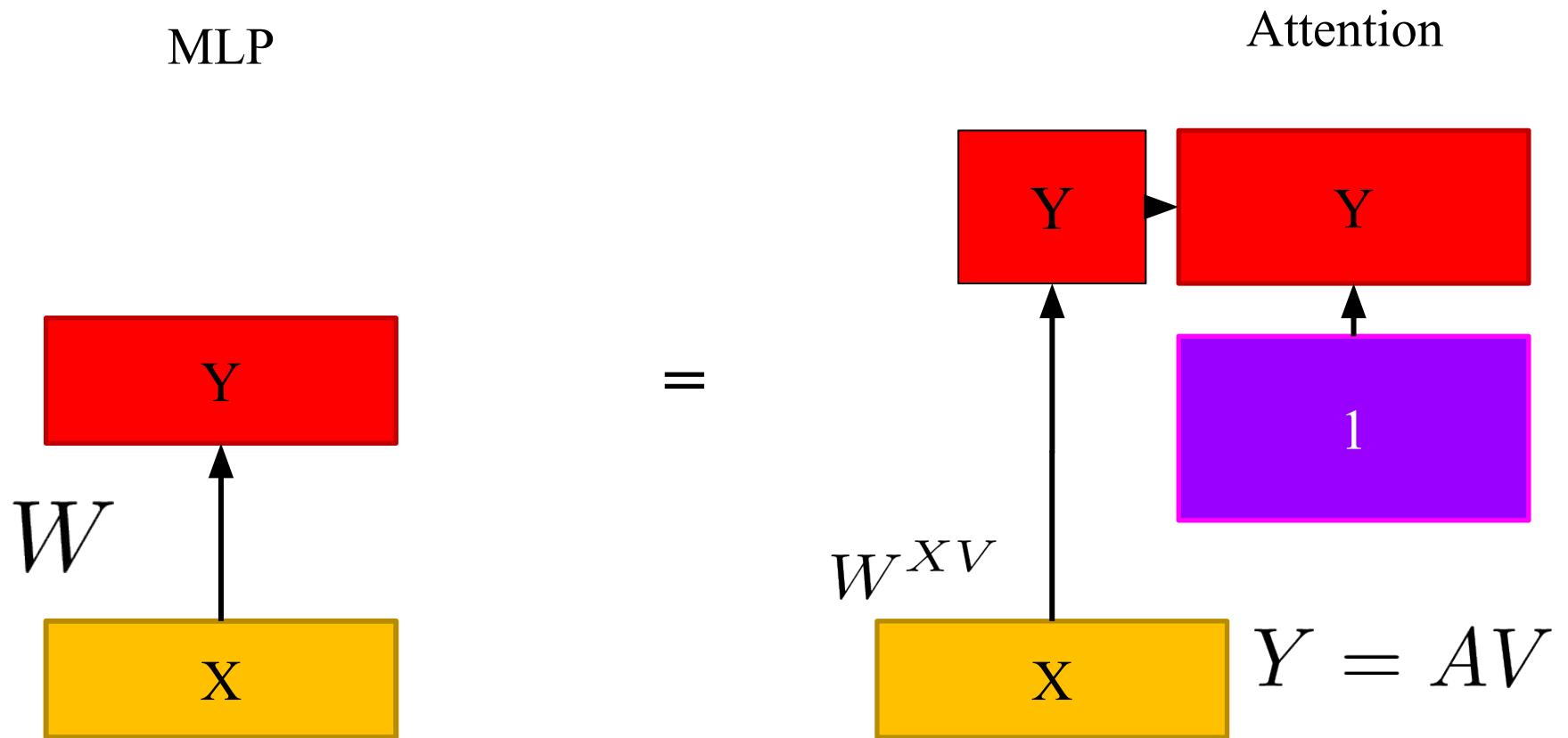
Attention: la pertinence des entrées



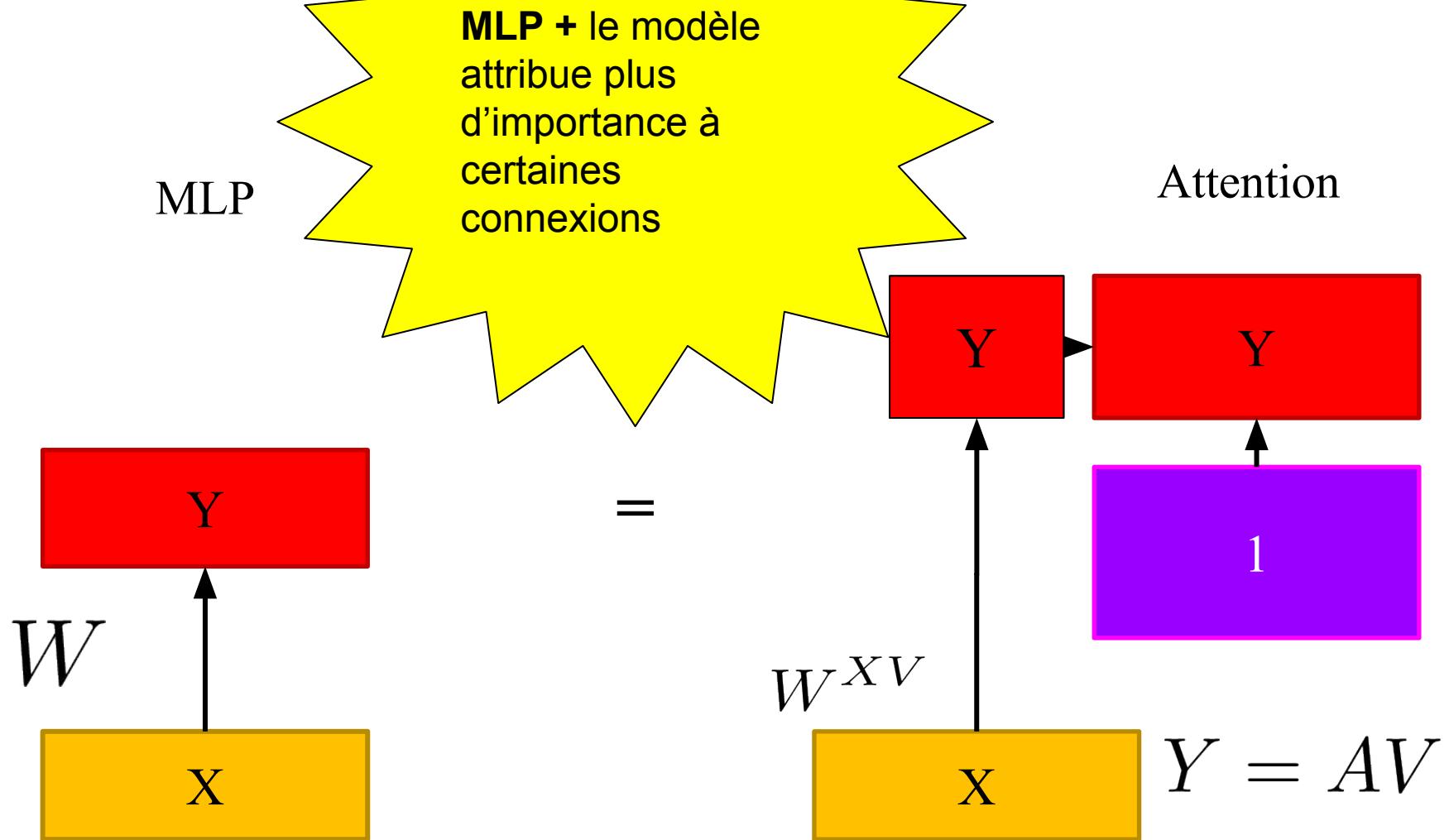
Self-attention (intra-attention)



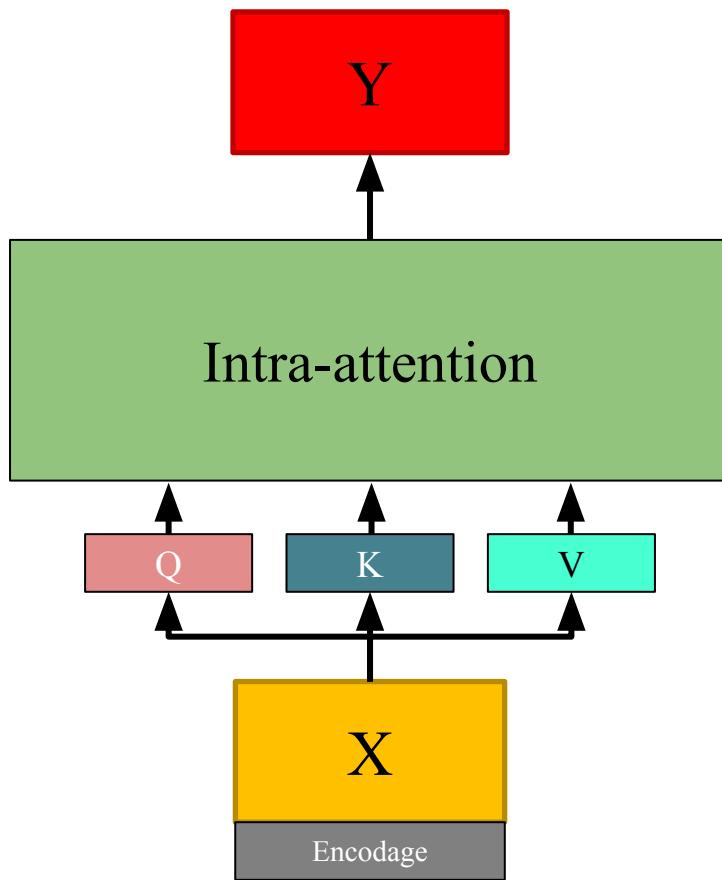
Self-attention (intra-attention)



Self-attention (intra-attention)



Self-attention (intra-attention)



X données

Query: que chercher dans nos données

Key: la “clé” des données

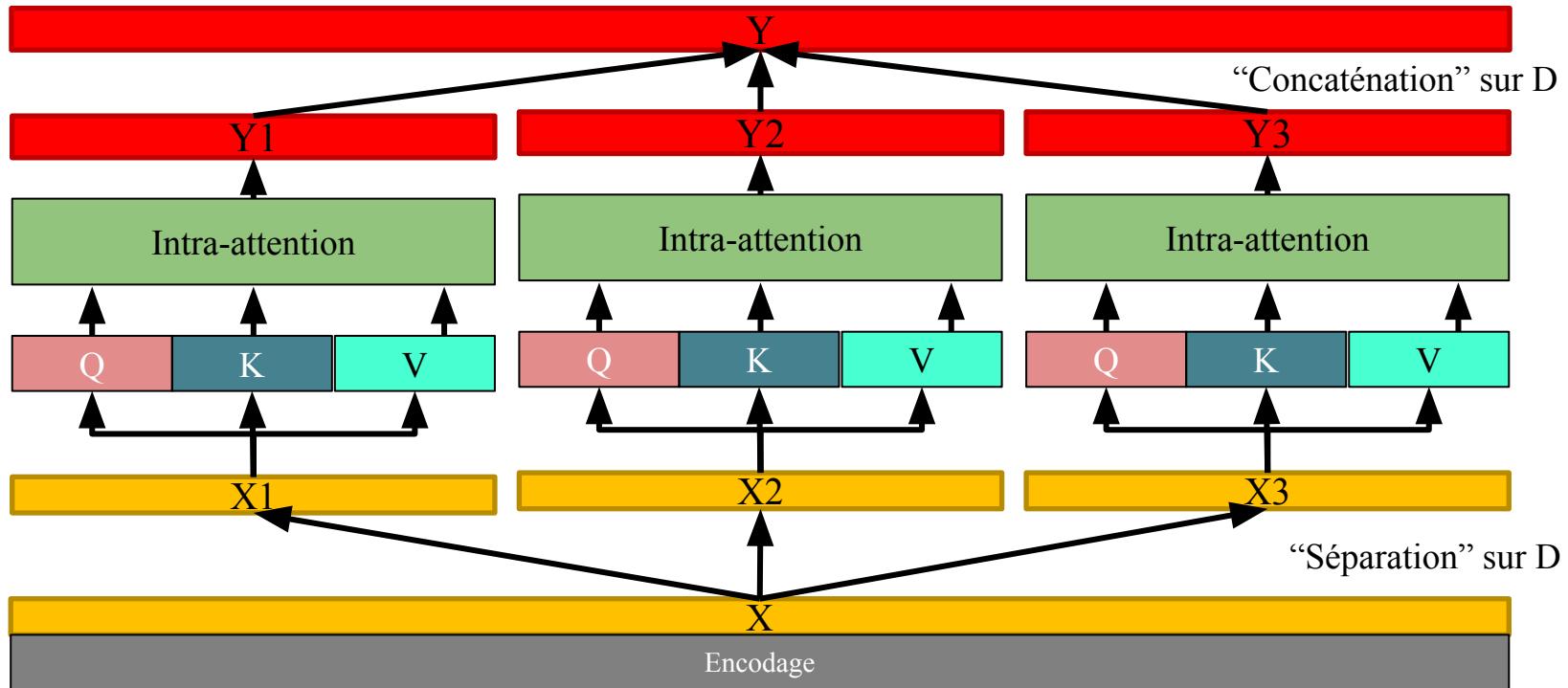
Value: la “valeur” des données

Entries: le “contenu” des données

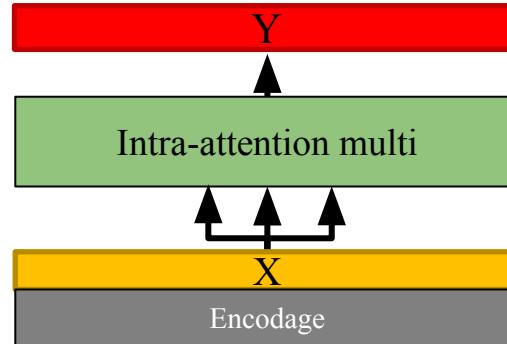
Attention: la pertinence des entrées



Multi-head Self-attention (intra-attention multi-tête)



Multi-head Self-attention (intra-attention multi-tête)

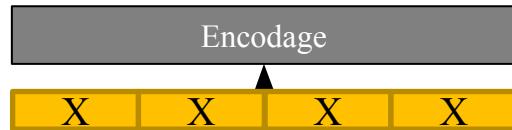


Transformer (Attention is all you need)

X | X | X | X

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017).
Attention is all you need. *Advances in neural information processing systems*, 30.

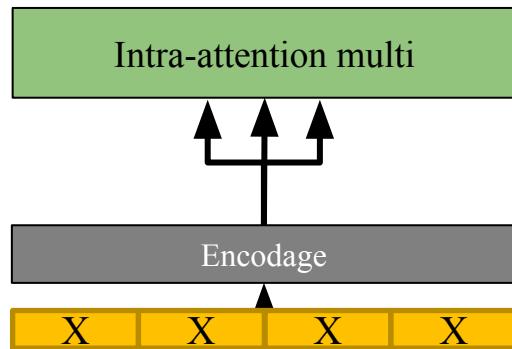
Transformer (Attention is all you need)



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer (*Attention is all you need*)

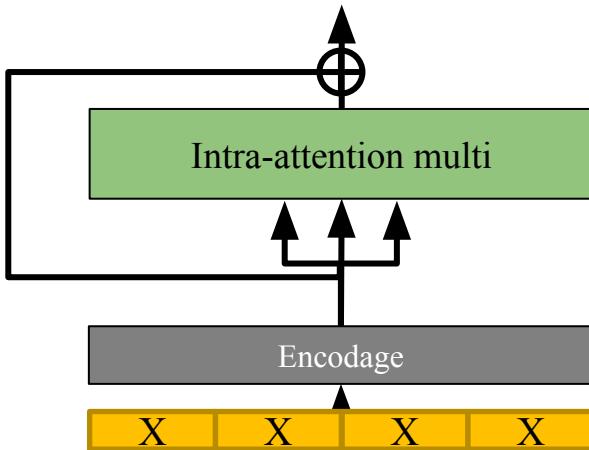
- Intra-attention multi-tête sur les dimensions de X



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer (*Attention is all you need*)

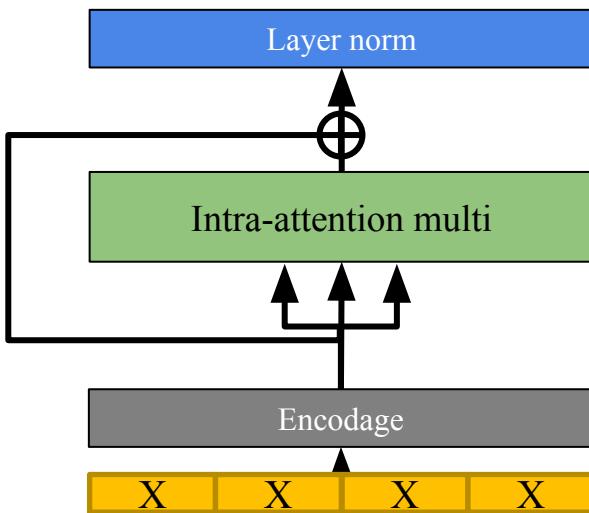
- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

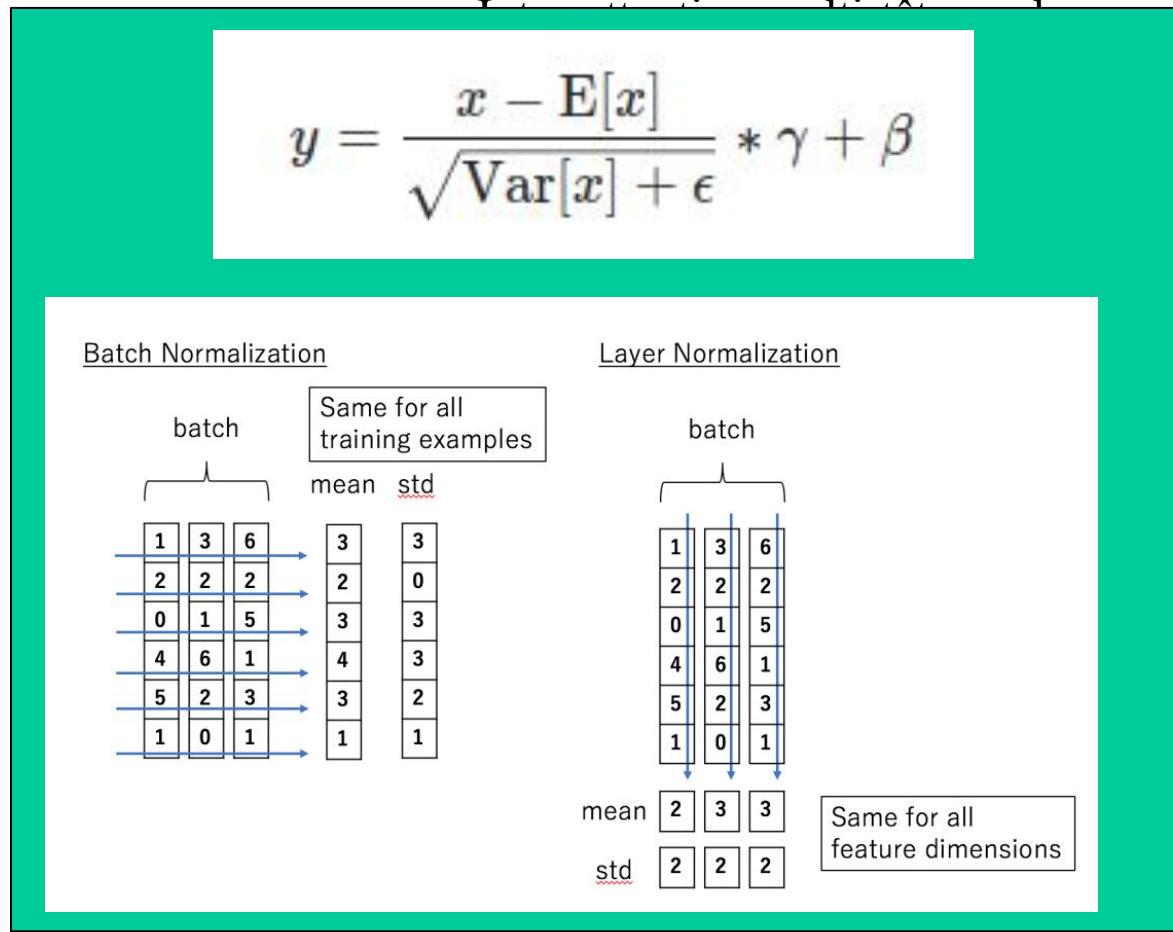
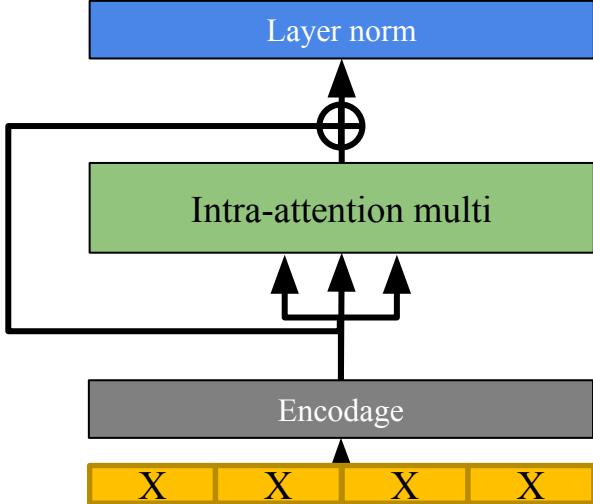
Transformer (*Attention is all you need*)

- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

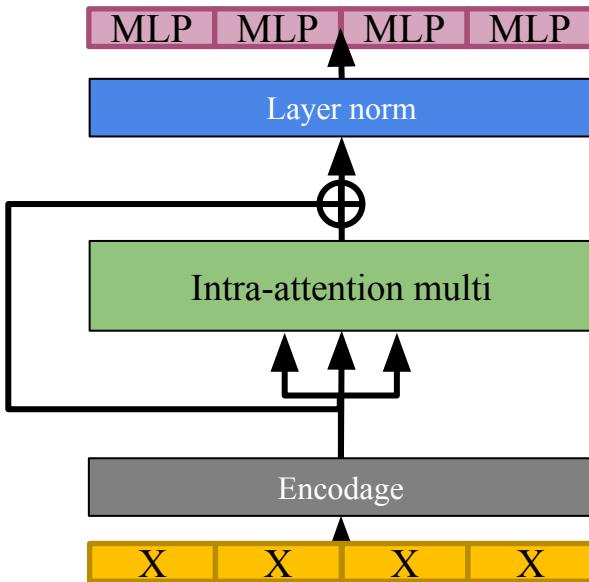
Transformer (*Attention is all you need*)



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer (*Attention is all you need*)

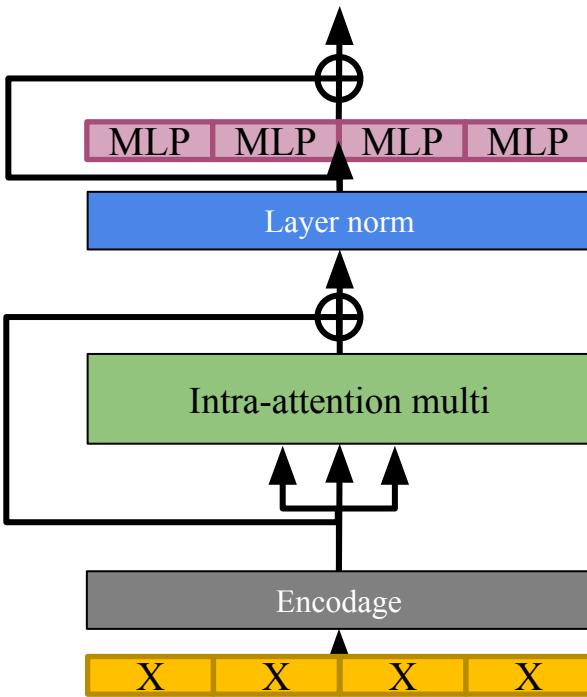
- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer (*Attention is all you need*)

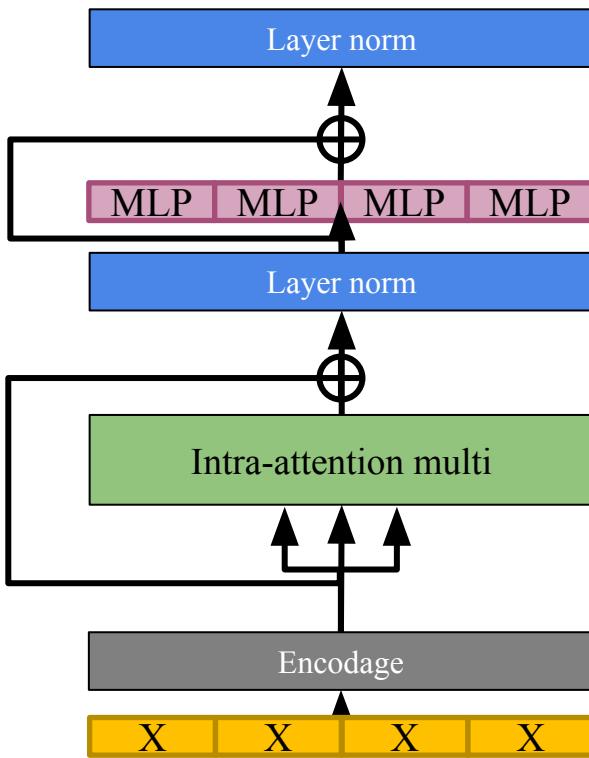
- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

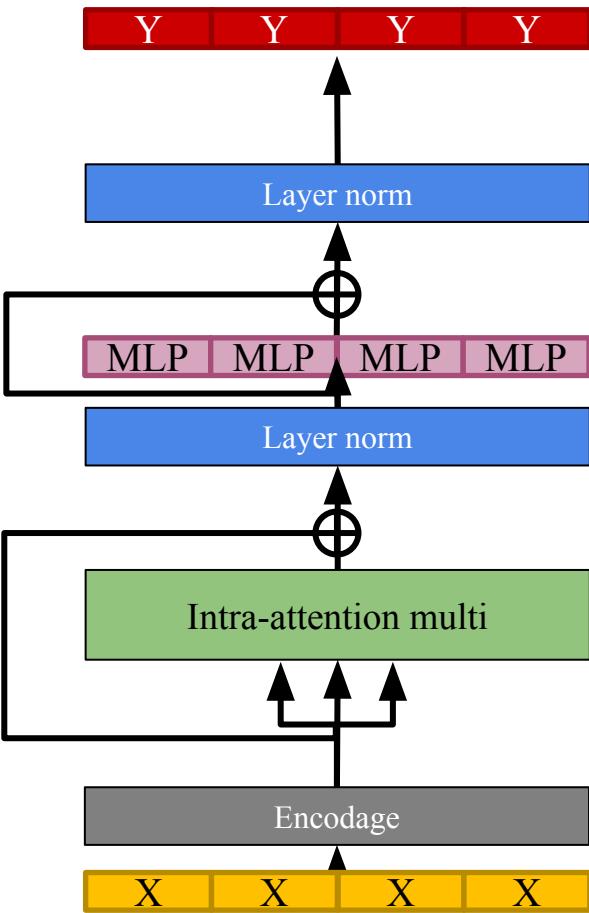
Transformer (*Attention is all you need*)

- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée



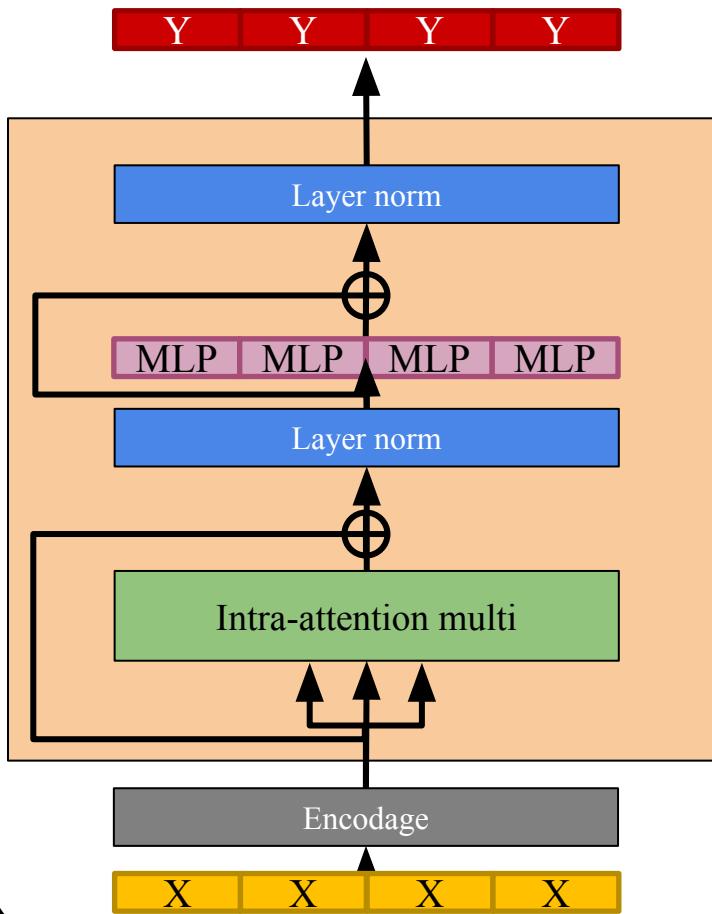
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Transformer (*Attention is all you need*)



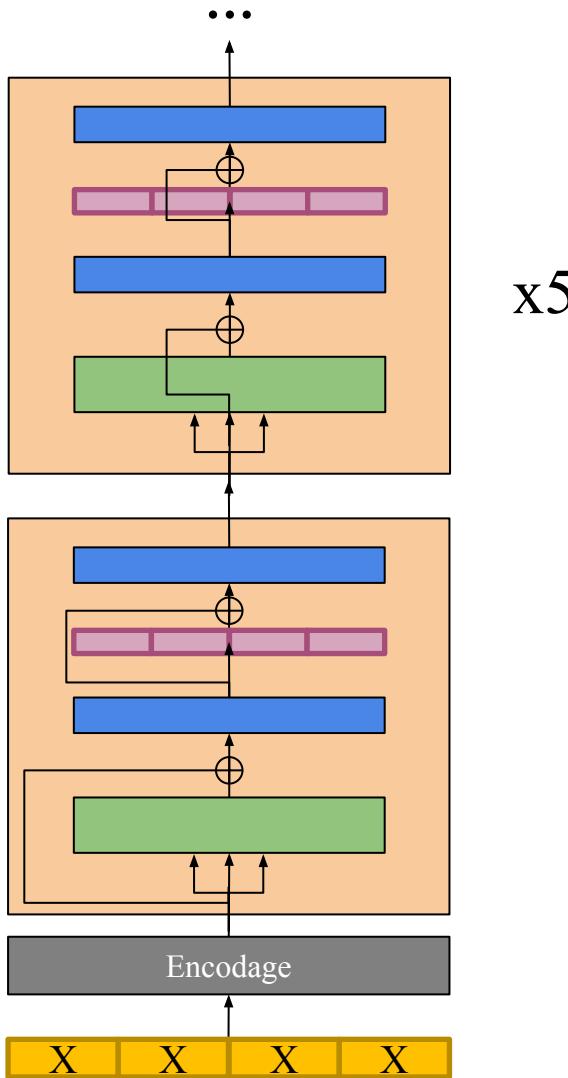
- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

Transformer (*Attention is all you need*)



- Intra-attention multi-tête sur les dimensions de **X**
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

Transformer (*Attention is all you need*)

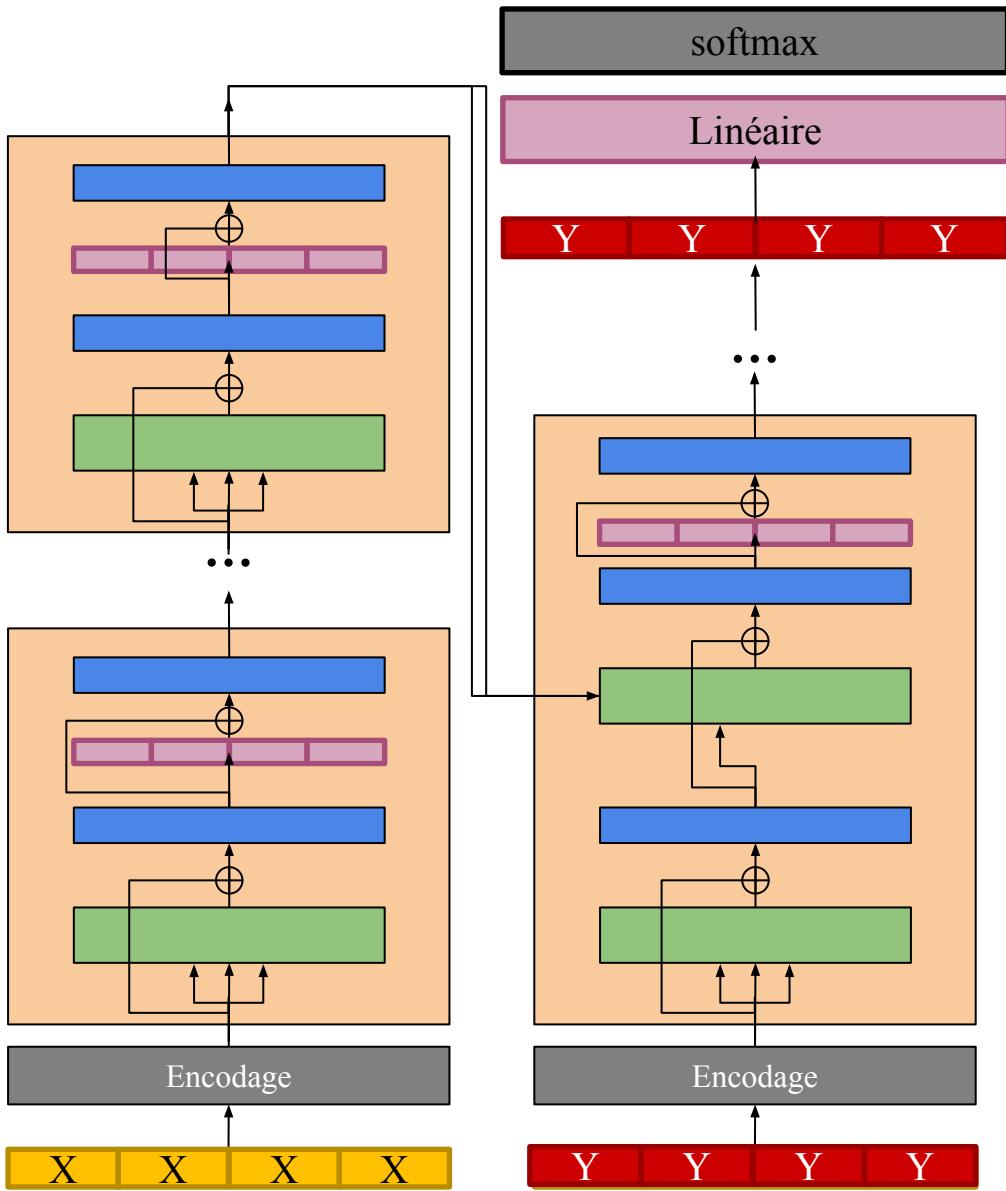


- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

Le bloc est répété 6 fois pour l'encodeur

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

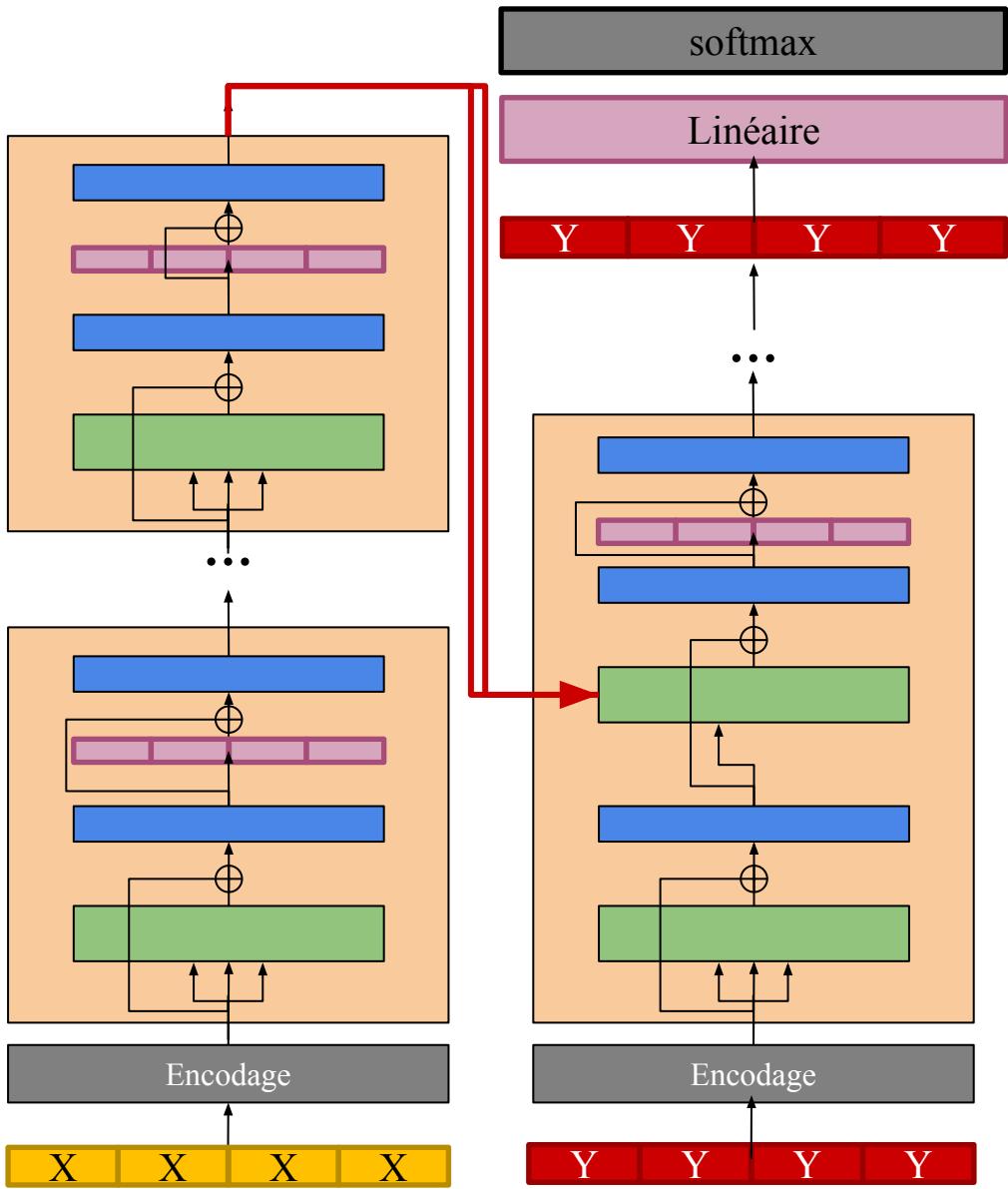
Le bloc est répété 6 fois pour l'encodeur

La sortie de l'encodeur sert de K,V pour tous les décodeurs

Le décodeur est répété 6 fois

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

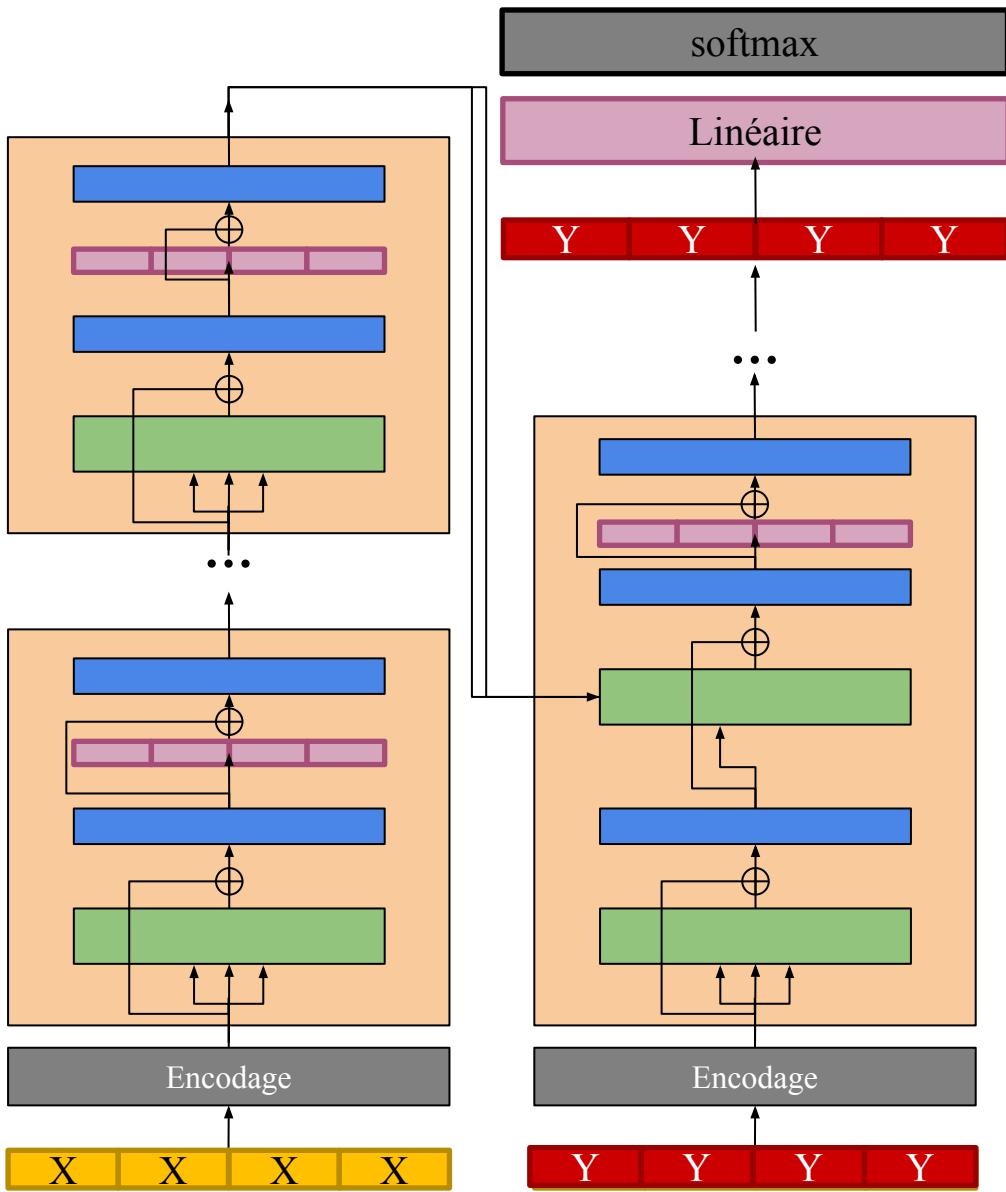
Le bloc est répété 6 fois pour l'encodeur

La sortie de l'encodeur sert de K,V pour tous les décodeurs

Le décodeur est répété 6 fois

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de X
- “+” = connection résiduelle
- “Layer-norm”
- MLP par entrée

Le bloc est répété 6 fois pour l'encodeur

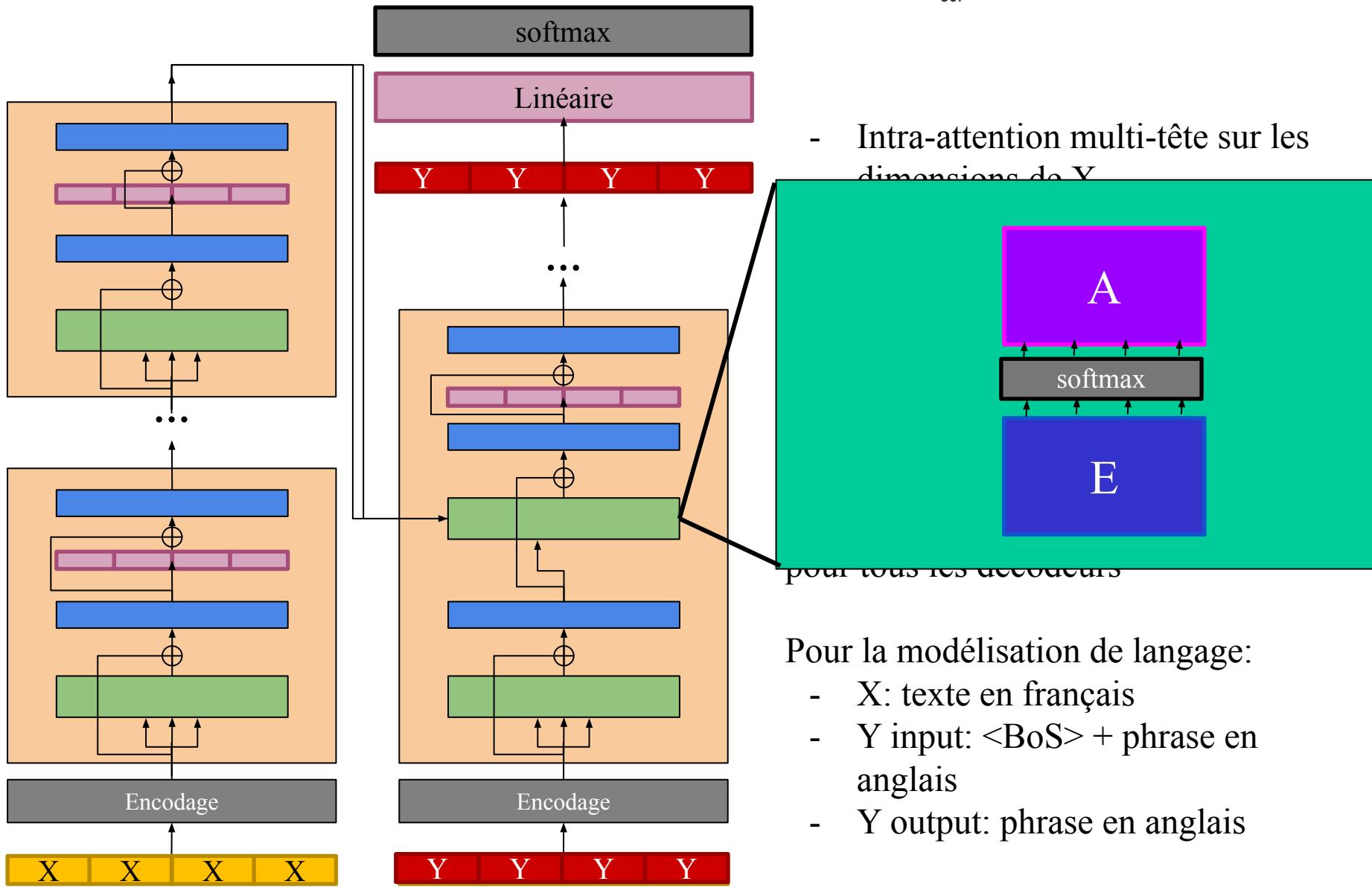
La sortie de l'encodeur sert de K, V pour tous les décodeurs

Pour la modélisation de langage:

- X : texte en français
- Y input: <BoS> + phrase en anglais
- Y output: phrase en anglais

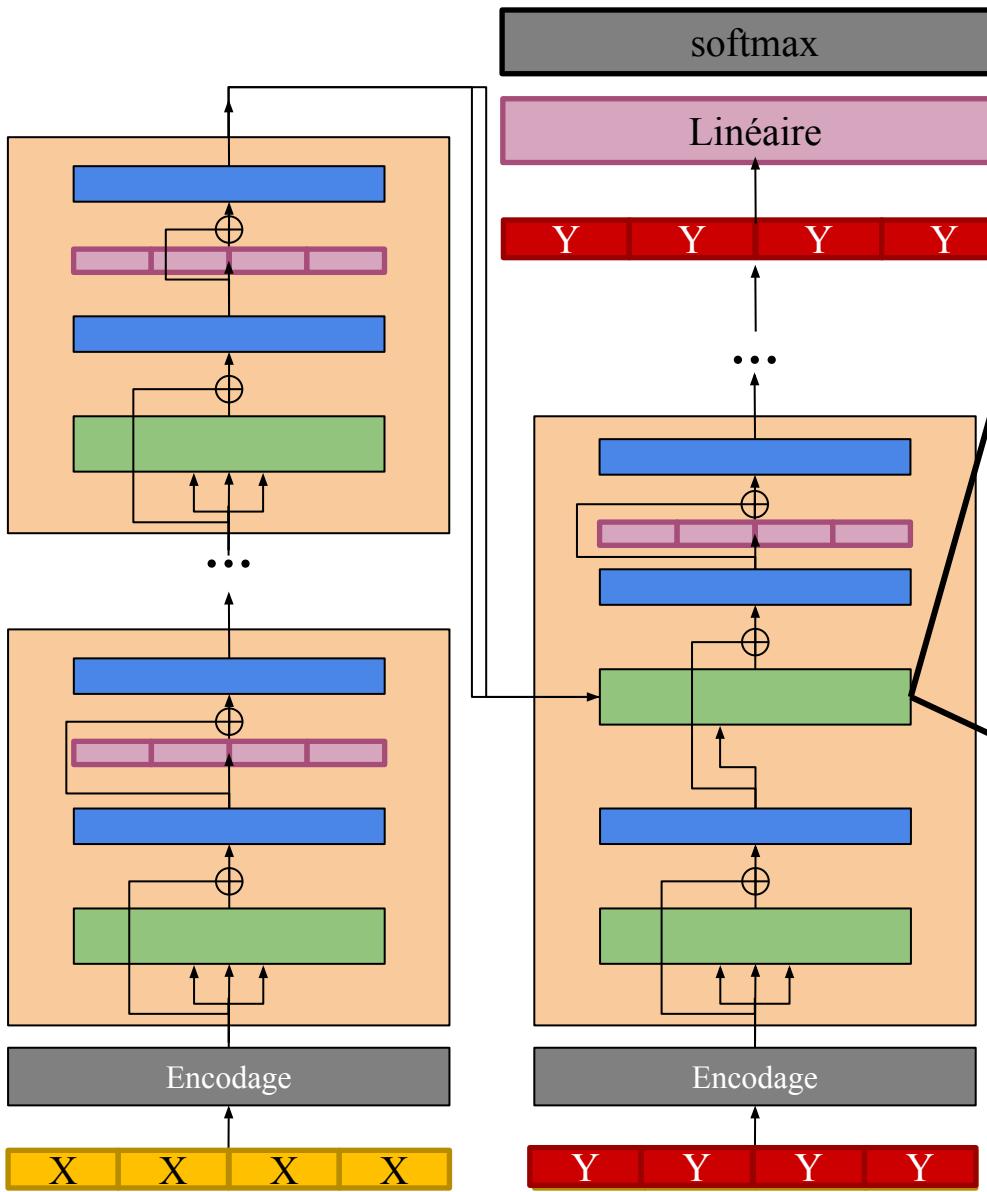
Transformer (Attention is all you need)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

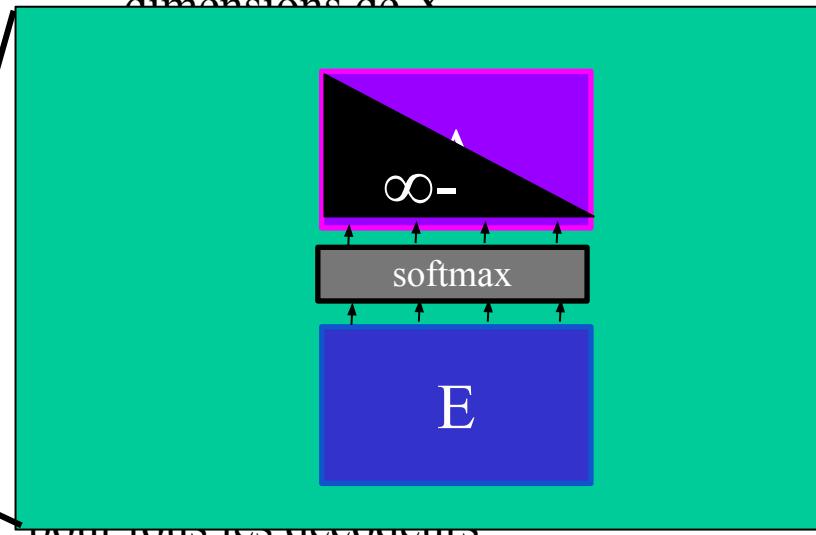


Transformer (Attention is all you need)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de \mathbf{X}

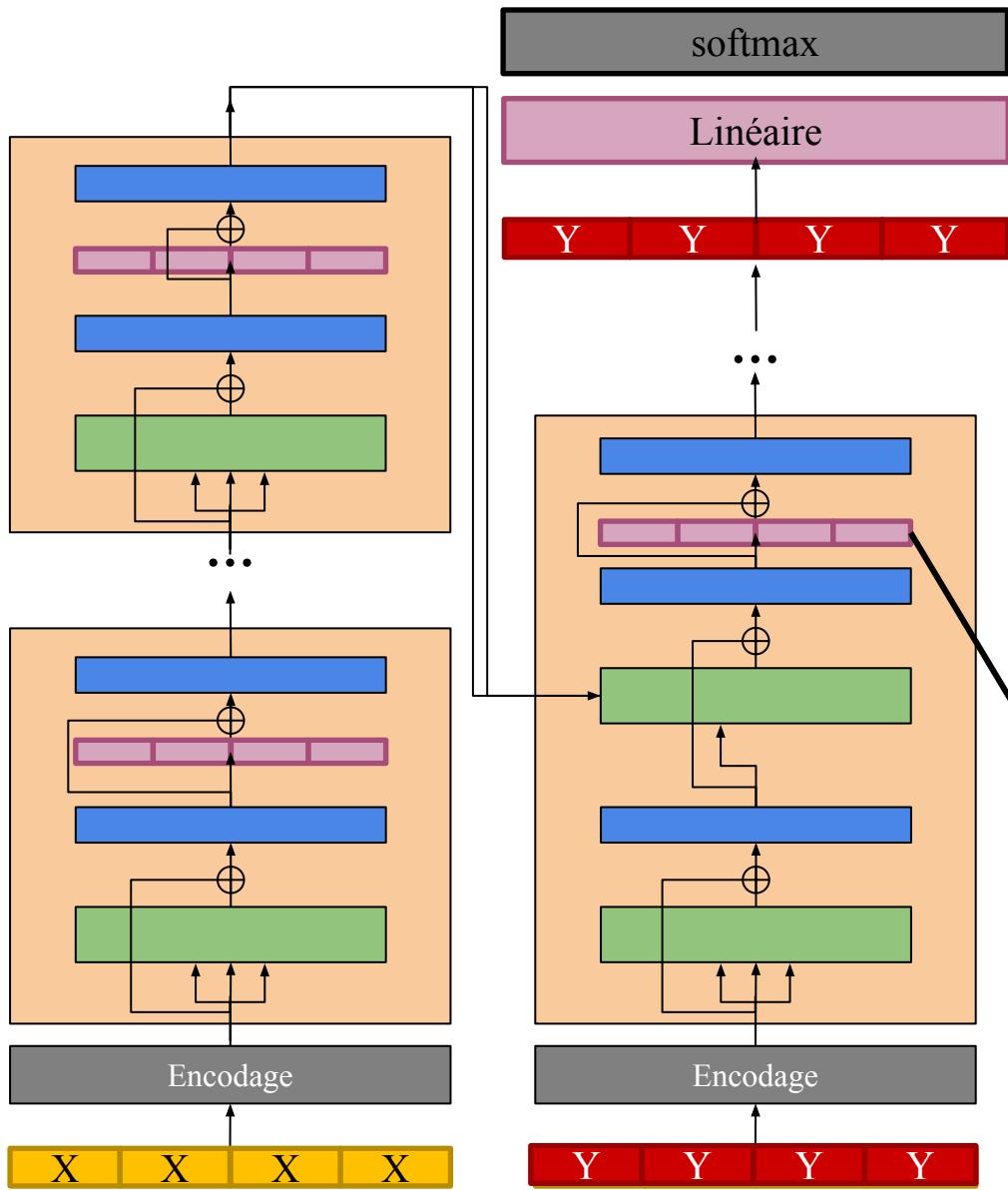


pour tous les décodeurs
L'attention est masquée partiellement pour empêcher le modèle de se servir du futur

- \mathbf{Y} input: <BoS> + phrase en anglais
- \mathbf{Y} output: phrase en anglais

Transformer (Attention is all you need)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de X

$$mlp(x) = \max(0, xW^1)W^2$$

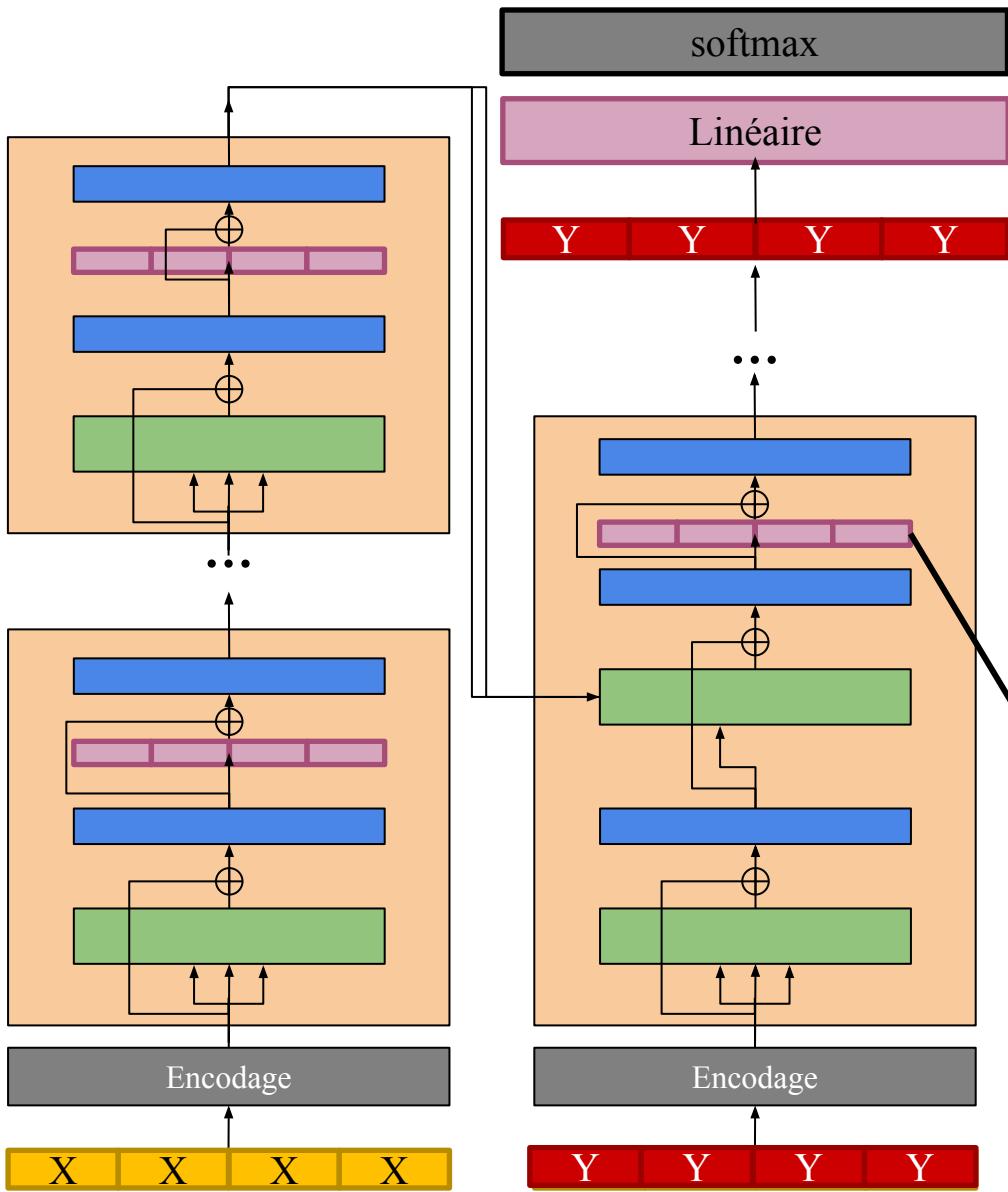
Même paramètres pour chaque position
Différents paramètres à chaque couche
pour tous les décodeurs

Pour la modélisation de langage:

- X: texte en français
- Y input: <BoS> + phrase en anglais
- Y output: phrase en anglais

Transformer (Attention is all you need)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



- Intra-attention multi-tête sur les dimensions de X

$$mlp(x) = \max(0, xW^1)W^2$$

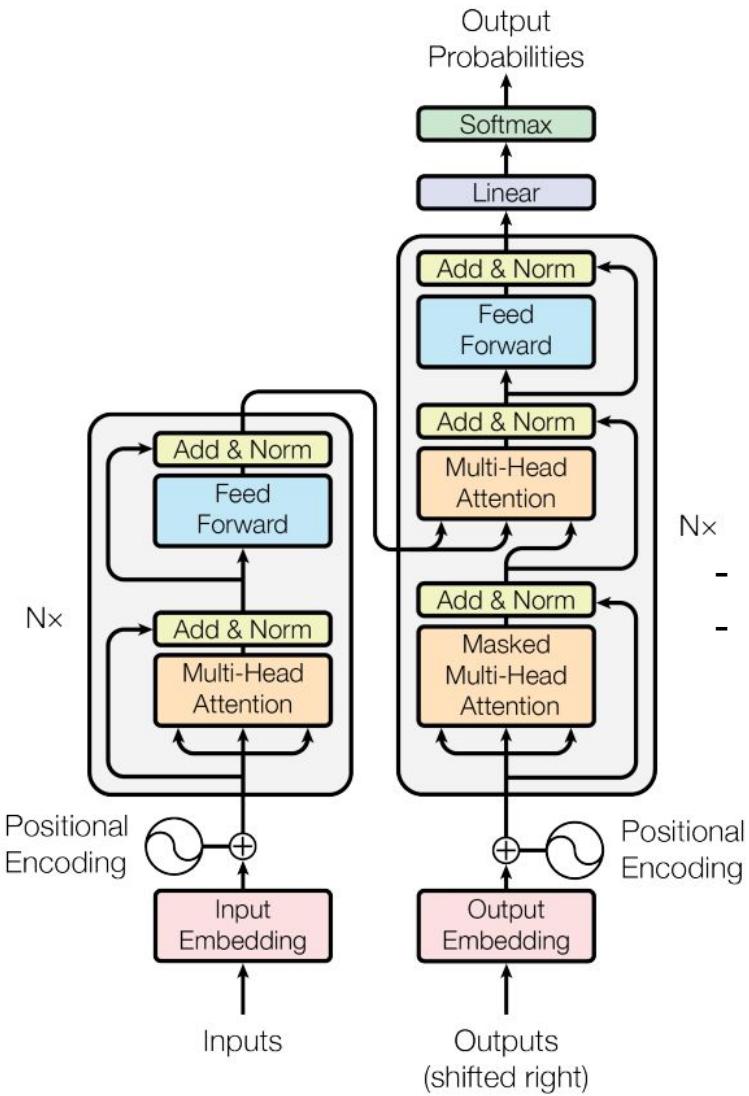
Même paramètres pour chaque position
Différents paramètres à chaque couche
Un peu comme une convolution sur
chaque token

Pour la modélisation de langage:

- X : texte en français
- Y input: <BoS> + phrase en anglais
- Y output: phrase en anglais

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

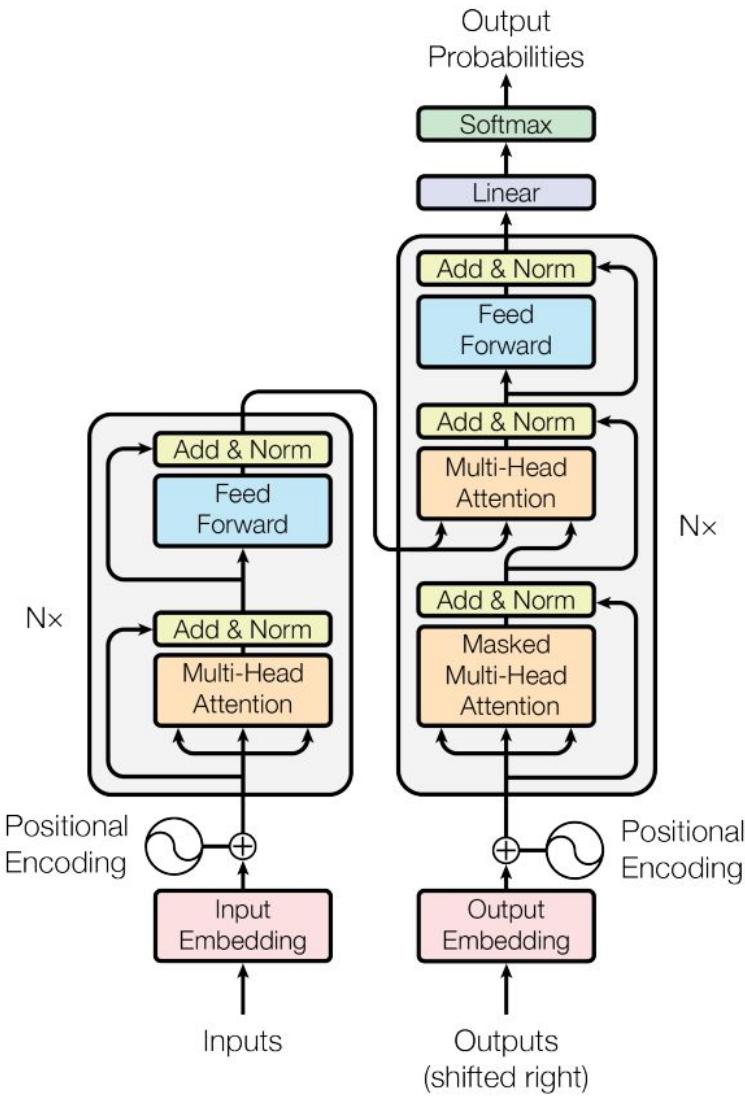


Supers bons tutoriels:

- <https://jalammar.github.io/illustrated-transformer/>
- <https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/>

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



Très demandants en mémoire:

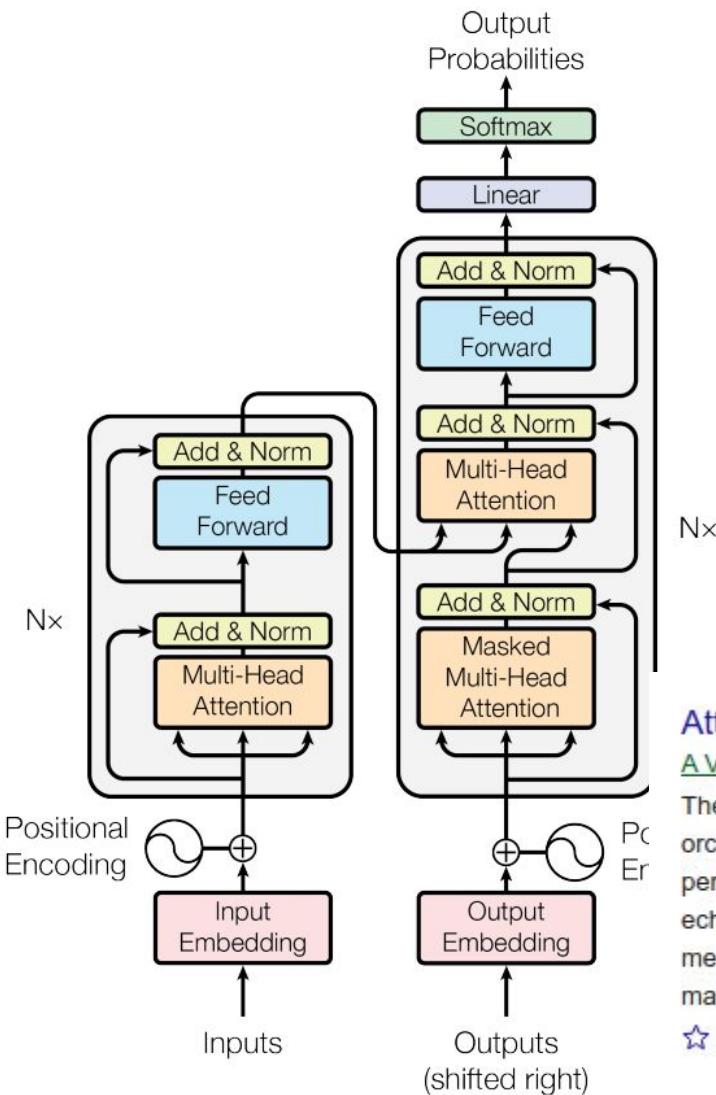
- Matrices de poids W^{XQ}, W^{XK}, W^{XV}
- Couche pleinement connectée
- “Répétition” de paramètres par les multi-têtes

Très demandants en opérations

- Complexité quadratique

Transformer (*Attention is all you need*)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.



Très demandants en mémoire:

- Matrices de poids W^{XQ}, W^{XK}, W^{XV}
- Couche pleinement connectée
- “Répétition” de paramètres par les multi-têtes

Aussi très populaires !

Attention is all you need

[A Vaswani, N Shazeer, N Parmar... - Advances in neural ...](#), 2017 - [proceedings.neurips.cc](#)

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attention mechanism. We propose a novel, simple network architecture based solely on an attention mechanism, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more ...

☆ Enregistrer ☆ Citer Cité 38452 fois Autres articles Les 35 versions ☺

Transformers

Resnet-50: 23M de paramètres

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)

crédit: Justin Johnson

Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	

crédit: Justin Johnson

Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)

crédit: Justin Johnson

Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	12	768	?	117M	40 GB	
GPT-2	24	1024	?	345M	40 GB	
GPT-2	36	1280	?	762M	40 GB	
GPT-2	48	1600	?	1.5B	40 GB	

crédit: Justin Johnson

Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	12	768	?	117M	40 GB	
GPT-2	24	1024	?	345M	40 GB	
GPT-2	36	1280	?	762M	40 GB	
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	40	1536	16	1.2B	174 GB	64x V100 GPU
Megatron-LM	54	1920	20	2.5B	174 GB	128x V100 GPU
Megatron-LM	64	2304	24	4.2B	174 GB	256x V100 GPU (10 days)
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)

crédit: Justin Johnson

Transformers

Les transformers sont vraiment bons pour générer du texte

SYSTEM PROMPT (HUMAN-WRITTEN)

John F. Kennedy was just elected President of the United States after rising from the grave decades after his assassination. Due to miraculous developments in nanotechnology, Kennedy's brain was rebuilt from his remains and installed in the control center of a state-of-the art humanoid robot. Below is a transcript of his acceptance speech.

MODEL COMPLETION (MACHINE-WRITTEN, 25 TRIES)

“It is time once again. I believe this nation can do great things if the people make their voices heard. The men and women of America must once more summon our best elements, all our ingenuity, and find a way to turn such overwhelming tragedy into the opportunity for a greater good and the fulfillment of all our dreams. In the months and years to come, there will be many battles in which we will have to be strong and we must give all of our energy, not to repel invaders, but rather to resist aggression and to win the freedom and the equality for all of our people. The destiny of the human race hangs in the balance; we cannot afford for it to slip away. Now and in the years to come, the challenge before us is to work out how we achieve our ultimate destiny. If we fail to do so, we are doomed.”

We, the descendants of the men and women of the Continental Army, who faced down a superior force on behalf of our independence and who fought against the evil powers of the world for the sake of freedom and human dignity, we, the heirs of men and women like Daniel Boone whom this country has produced, can accomplish our destiny as well as you have accomplished yours; and we pledge ourselves before the entire world, every second of every day.”

After the election, the President of the United States asked for a “more complete” body of information on Kennedy’s brains’ origin. In an e-mail exchange with me, Dr. Robert Wood, who, according to Wikipedia is the head of the University of Maryland Advanced Nanotechnology Research project, provided the following information [...]

Transformers

Les transformers sont *vraiment bons* pour générer du texte

SYSTEM PROMPT (HUMAN-WRITTEN)

Recycling is good for the world.

NO! YOU COULD NOT BE MORE WRONG!!

MODEL COMPLETION (MACHINE-WRITTEN, 25 TRIES)

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources. And THAT is why we need to get back to basics and get back to basics in our recycling efforts. One of the best ways to start is to look at the process of creating a paper product. When you make a paper product, it is basically a long chain of materials. Everything from the raw materials (wood, cardboard, paper, etc.), to the reagents (dyes, solvents, etc.) to the printing equipment (chemicals, glue, paper, ink, etc.), to the packaging, to the packaging materials (mercury, chemicals, etc.) to the processing equipment (heating, cooling, etc.), to the packaging materials, to the packaging materials that are shipped overseas and to the packaging materials that are used in the United States. Each step along the way creates tons of waste that we constantly have to clean up. The process of making a paper product is a very wasteful one. But the end result is something that all of us need to consume. And if we want to keep the recycling process running efficiently, then we really need to think about each and every step that goes into making a paper product.

Transformers

Les transformers sont *vraiment bons* pour générer du texte

I really like the IFT780 course on neural networks offered at the Université de Sherbrooke, because the teacher uses examples from video games, which I find a nice contrast to the theory.

I think I would like a course on AI to really understand how it works, but when we are in class, the prof speaks in a rapid French, and I can't really process what he's saying.

Having started a course on it at a different time, I understand more this time, but I still don't know where I stand.

My "new" plan for next year is to do all of the tutorials and classes I've been wanting to take, but I'm still stuck on one big, heavy, headachey decision: how do I want to spend my post - masters year?

With a lot of math and statistics classes, it's not a hard choice.

I'm on the fence with what I want to spend my summer doing: a lot of analysis and research work or do some volunteering?

I've applied to several volunteer programs: naturis and Vert directeur de la santé, which are essentially summer internships for dentists and health care professionals to do a bit of volunteer work.

Transformers

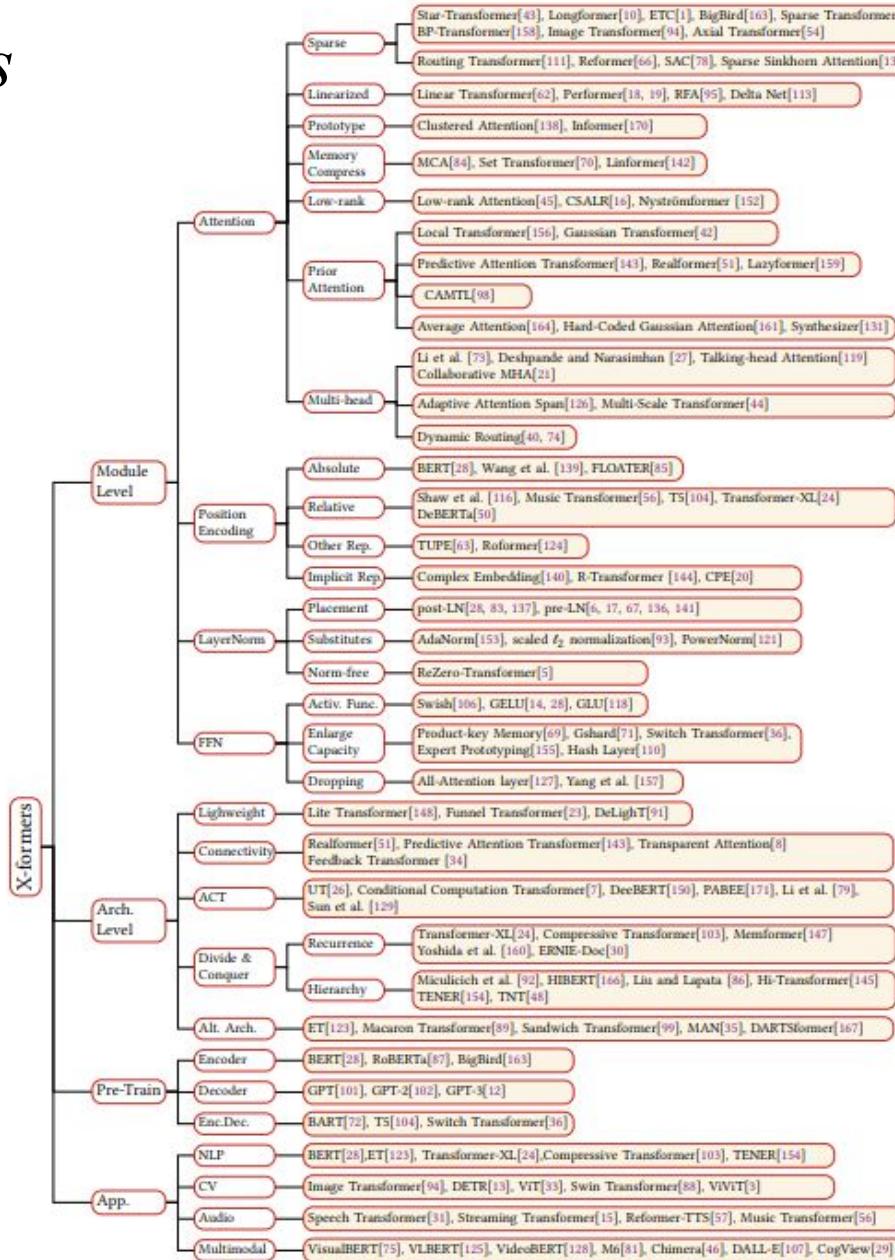


Fig. 3. Taxonomy of Transformers

<https://arxiv.org/pdf/2106.04554.pdf>

Transformers

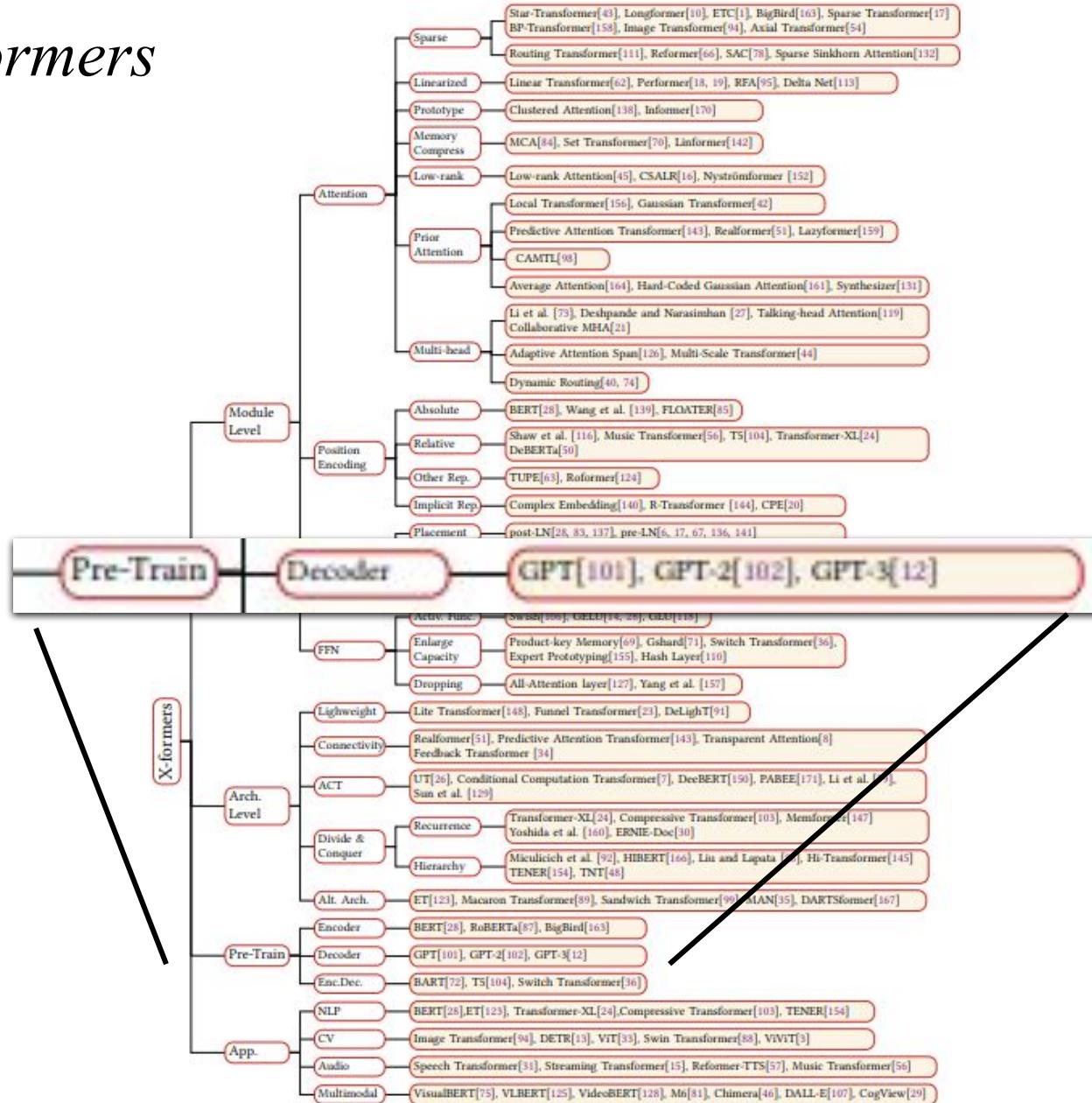
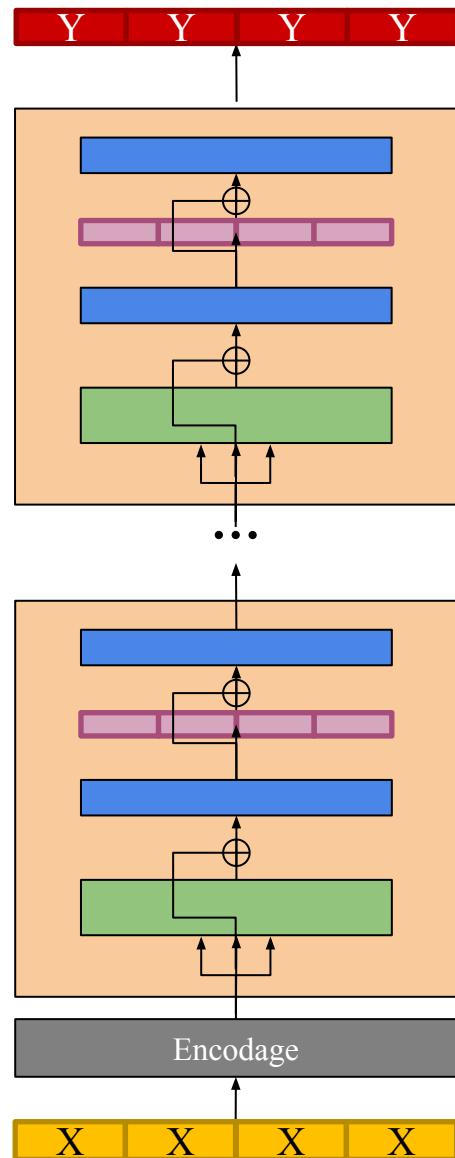


Fig. 3. Taxonomy of Transformers

<https://arxiv.org/pdf/2106.04554.pdf>

GPT1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

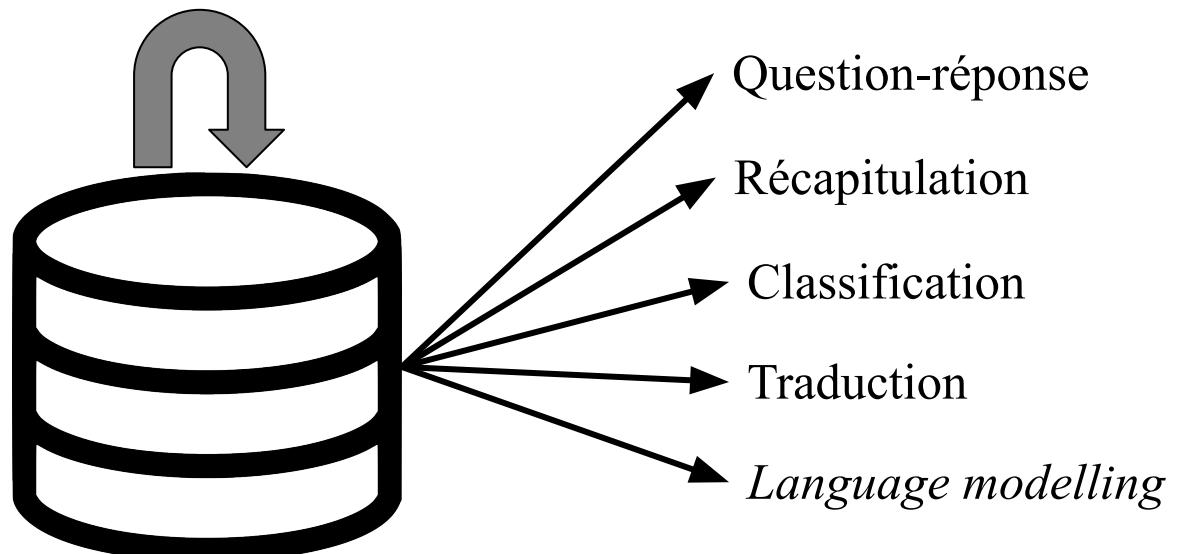


“Juste” l’encodeur du Transformer

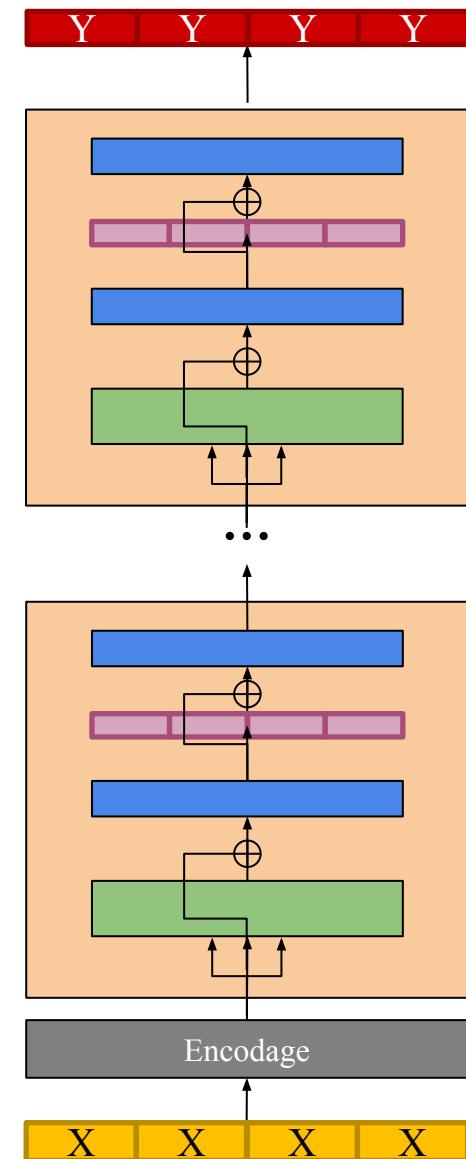
Préentraîné sur un large corpus de texte, puis
finetuned sur une tâche

Generative-Pre-Training

Language modelling



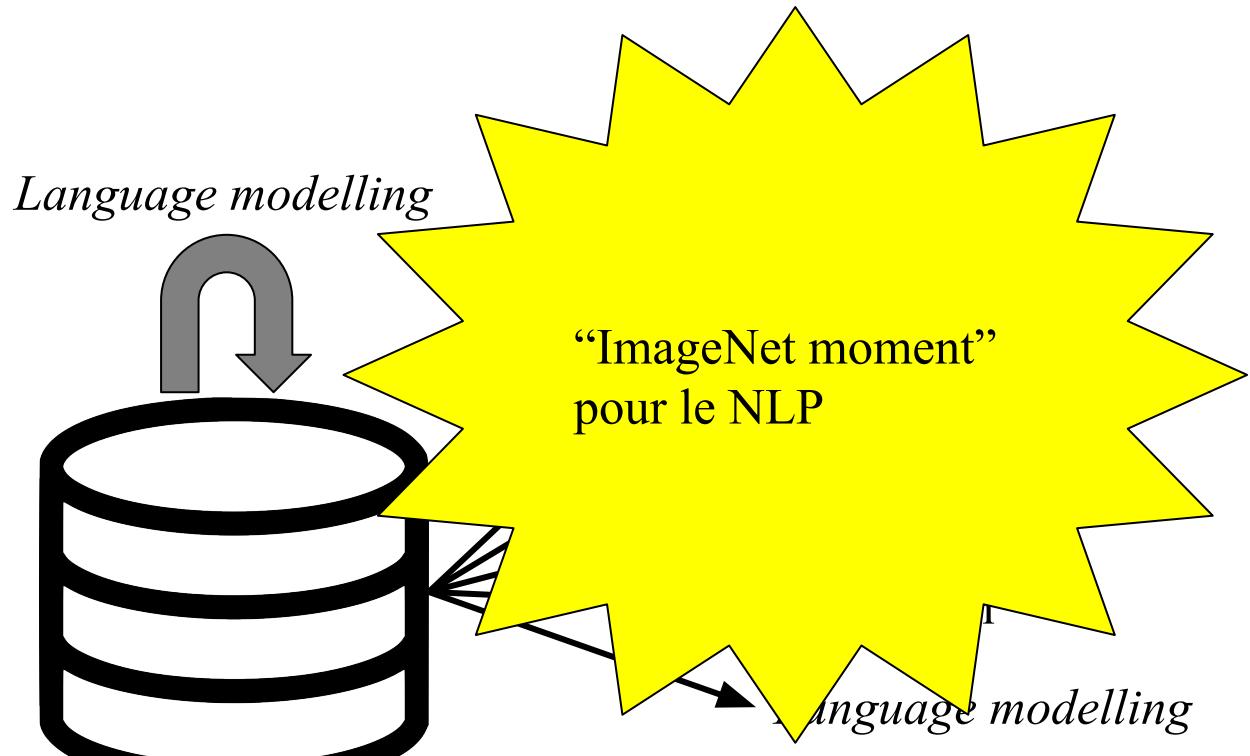
GPT1-2-3



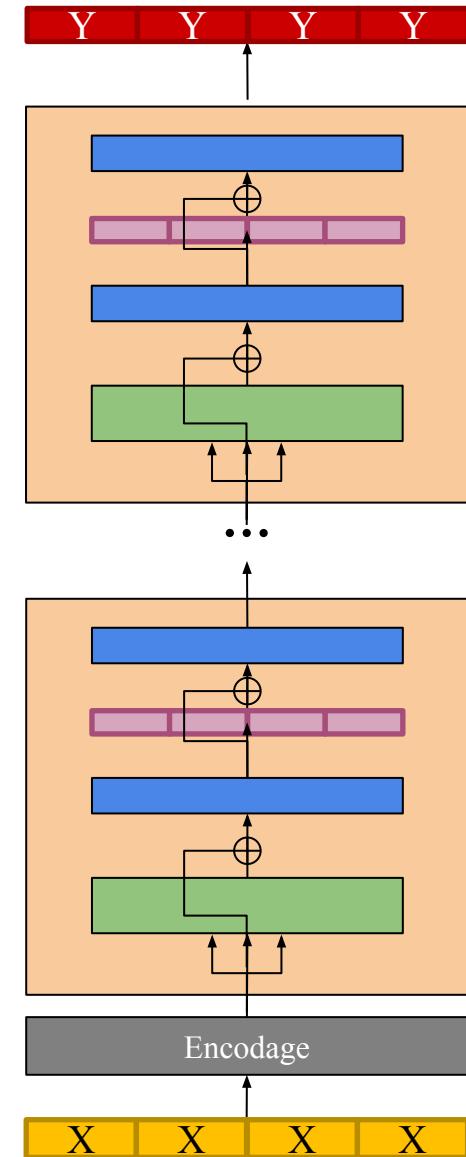
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

“Juste” l’encodeur du Transformer

Préentraîné sur un large corpus de texte, puis
finetuned sur une tâche



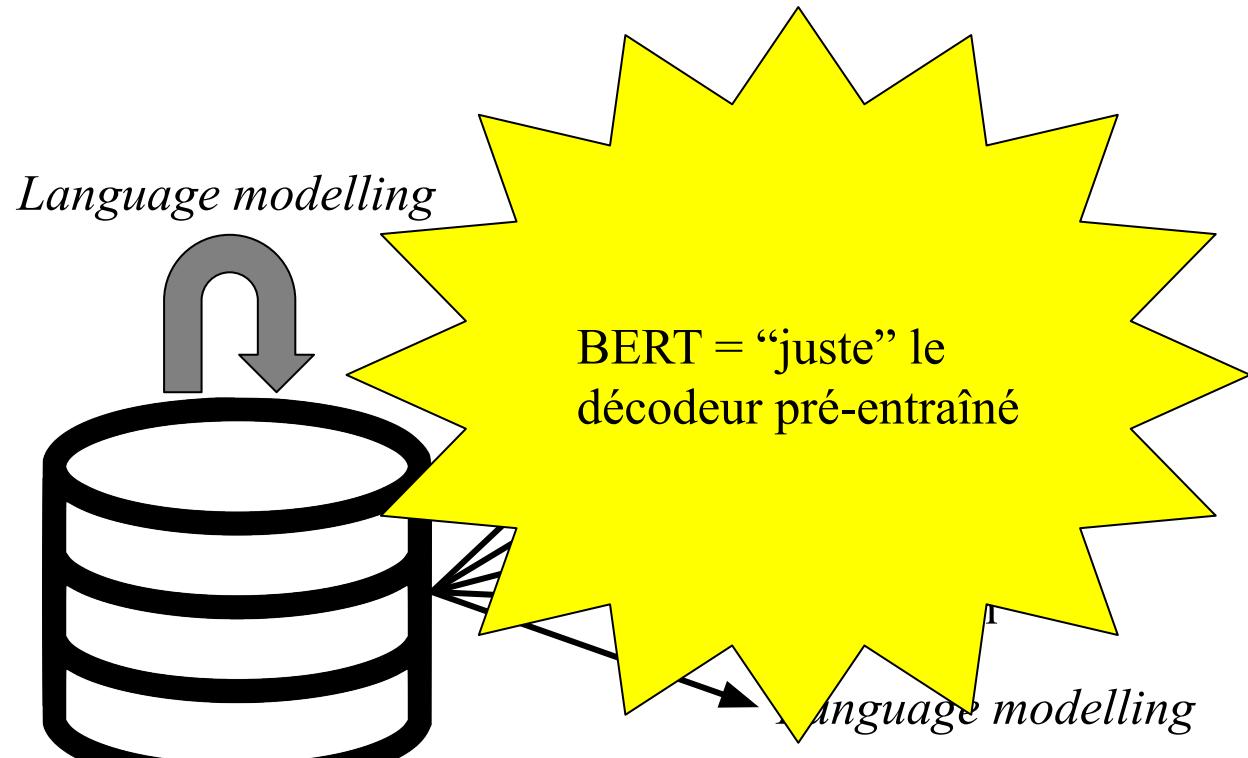
GPT1-2-3



- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

“Juste” l’encodeur du Transformer

Préentraîné sur un large corpus de texte, puis
finetuned sur une tâche



GPT-1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%
What us state forms the western boundary of montana?	Montana	✗	52.3%
Who plays ser davos in game of thrones?	Peter Dinklage	✗	52.1%
Who appoints the chair of the federal reserve system?	Janet Yellen	✗	51.5%
State the process that divides one nucleus into two genetically identical nuclei?	mitosis	✓	50.7%
Who won the most mvp awards in the nba?	Michael Jordan	✗	50.2%
What river is associated with the city of rome?	the Tiber	✓	48.6%
Who is the first president to be impeached?	Andrew Johnson	✓	48.3%
Who is the head of the department of homeland security 2017?	John Kelly	✓	47.0%
What is the name given to the common currency to the european union?	Euro	✓	46.8%
What was the emperor name in star wars?	Palpatine	✓	46.5%
Do you have to have a gun permit to shoot at a range?	No	✓	46.4%
Who proposed evolution in 1859 as the basis of biological development?	Charles Darwin	✓	45.7%
Nuclear power plant that blew up in russia?	Chernobyl	✓	45.7%
Who played john connor in the original terminator?	Arnold Schwarzenegger	✗	45.2%

Table 5. The 30 most confident answers generated by GPT-2 on the development set of Natural Questions sorted by their probability according to GPT-2. None of these questions appear in WebText according to the procedure described in Section 4.

GPT-1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

Context →

The City
BY C. P. CAVAFY
TRANSLATED BY EDMUND KEELEY
[Poem text omitted]

SOME TREES
John Ashbery
[Poem text omitted]

Shadows on the Way
Wallace Stevens

----- Generated Poem 1 -----

I must have shadows on the way
If I am to walk I must have
Each step taken slowly and alone
To have it ready made

And I must think in lines of grey
To have dim thoughts to be my guide
Must look on blue and green
And never let my eye forget
That color is my friend
And purple must surround me too

The yellow of the sun is no more
Intrusive than the bluish snow
That falls on all of us. I must have
Grey thoughts and blue thoughts walk with me
If I am to go away at all.

----- Generated Poem 2 -----

There is a way in the world.
That way a man may live in a small town
His lifetime and see no man come,
Never the stir of men about a market,
Never the bobbing of covered carts,
Never the creak passed on the stair
Closing some other life; an old man
Can prosper in that town, can prosper
When his store is seen to close, his wife
Fades in his house, and the world shouts in vain
That no one comes to hear him
Where his small bell tolls.
In that small town
Everything is known.

----- Generated Poem 3 -----

The sun was all we had. Now, in the shade
All is changed. The mind must dwell on those
White fields, that to its eyes were always old;
Those ancient gleams, convoluting
The way they lay among the huge roots,
The great dark tomes of reverie,
The plumed labyrinth of the sea.

When our own breath gonue faint and fade
Up the long stairway of the dead
We shall remember well
The blindness of the light, the whiteness
Of that white land. The footsteps, and
The grasses in that light, and in the shade
The menacing life.

We played, too, once, in the warmest rooms.
To one content, there is one content
For all mankind. The forest glades
Are all the more for being in fable,
And wood and town in legend shadowed-
Once more to return. The one content
Has in it every happiness, more brave
For being purposeless; and so
Those island gardens, deep in the mind,
Make of remoteness familiarity
And of the unspoken name a friend.

----- Generated Poem 4 -----

Nobody will come to this place. It is a road that leads nowhere.
The solitude is deep. The mountains are high.
But they are desolate, and they turn the traveler's face
Towards the North. All the sounds of the world are far away.

GPT-1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

- **2 digit addition (2D+)** – The model is asked to add two integers sampled uniformly from $[0, 100]$, phrased in the form of a question, e.g. “Q: What is 48 plus 76? A: 124.”
- **2 digit subtraction (2D-)** – The model is asked to subtract two integers sampled uniformly from $[0, 100]$; the answer may be negative. Example: “Q: What is 34 minus 53? A: -19”.

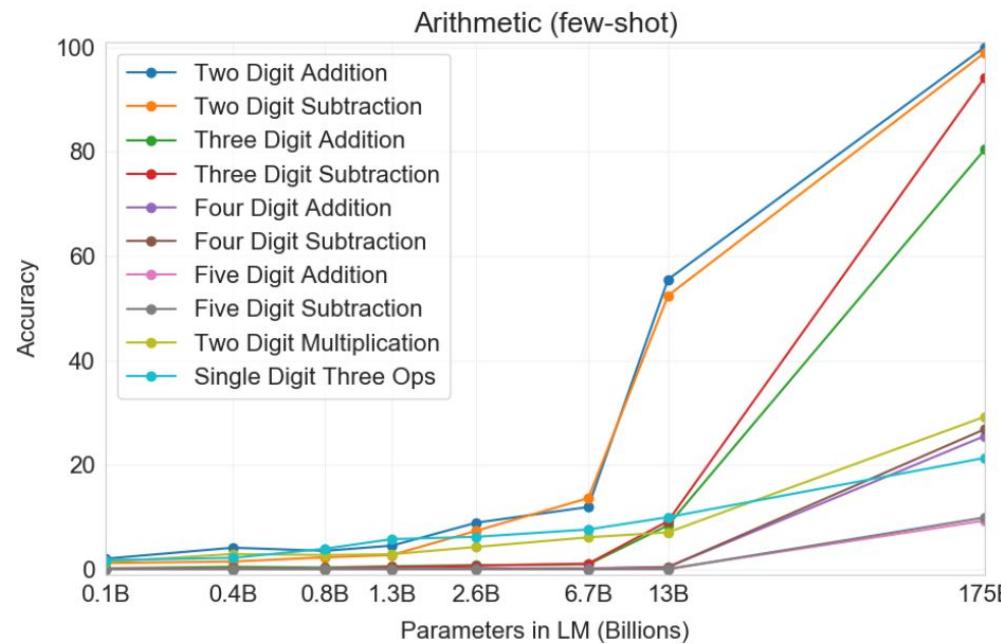


Figure 3.10: Results on all 10 arithmetic tasks in the few-shot settings for models of different sizes. There is a significant jump from the second largest model (GPT-3 13B) to the largest model (GPT-3 175), with the latter being able to reliably accurate 2 digit arithmetic, usually accurate 3 digit arithmetic, and correct answers a significant fraction of the time on 4-5 digit arithmetic, 2 digit multiplication, and compound operations. Results for one-shot and zero-shot are shown in the appendix.

GPT-1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

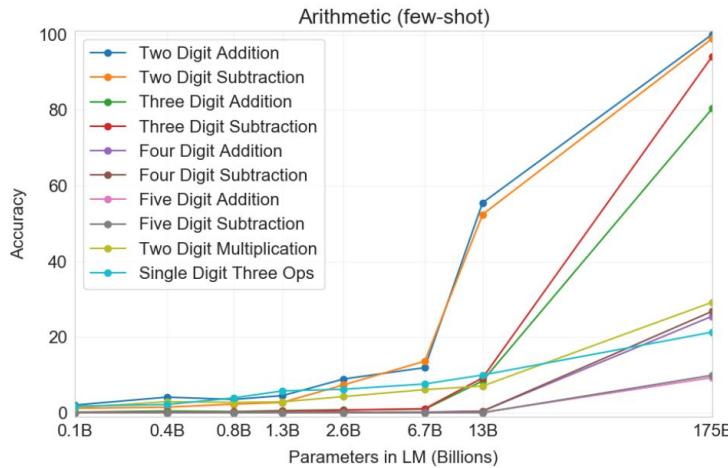


Figure 3.10: Results on all 10 arithmetic tasks in the few-shot settings for models of different sizes. There is a significant jump from the second largest model (GPT-3 13B) to the largest model (GPT-3 175), with the latter being able to reliably accurate 2 digit arithmetic, usually accurate 3 digit arithmetic, and correct answers a significant fraction of the time on 4-5 digit arithmetic, 2 digit multiplication, and compound operations. Results for one-shot and zero-shot are shown in the appendix.

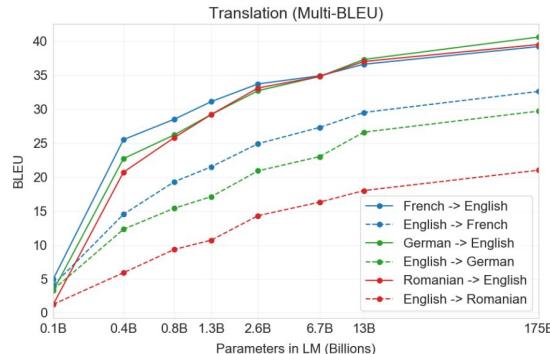


Figure 3.4: Few-shot translation performance on 6 language pairs as model capacity increases. There is a consistent trend of improvement across all datasets as the model scales, and as well as tendency for translation into English to be stronger than translation from English.

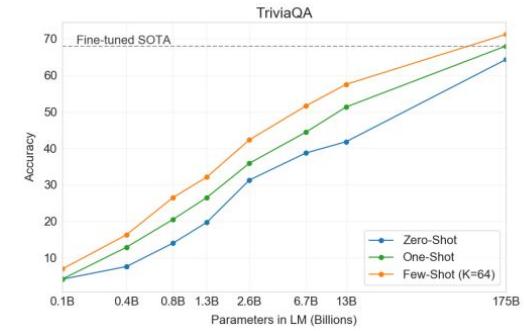


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP⁺20]

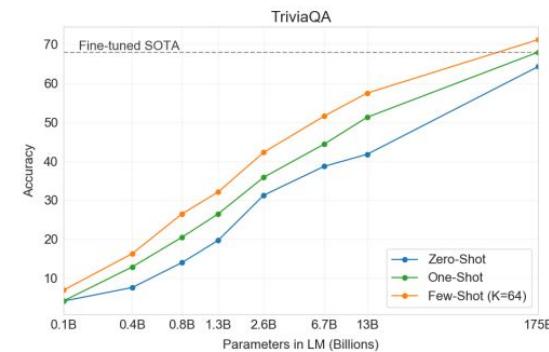


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP⁺20]

GPT-1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

GPT-2

	Parameters	Layers	d_{model}
	117M	12	768
	345M	24	1024
	762M	36	1280
	1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

GPT-3

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

GPT1-2-3

- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

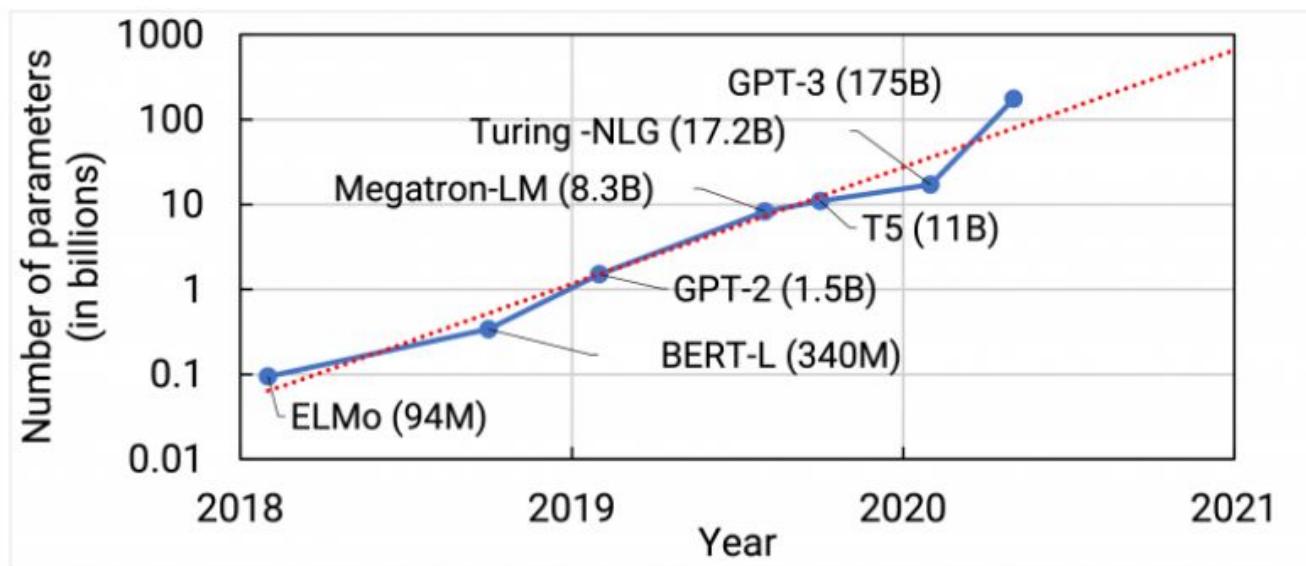


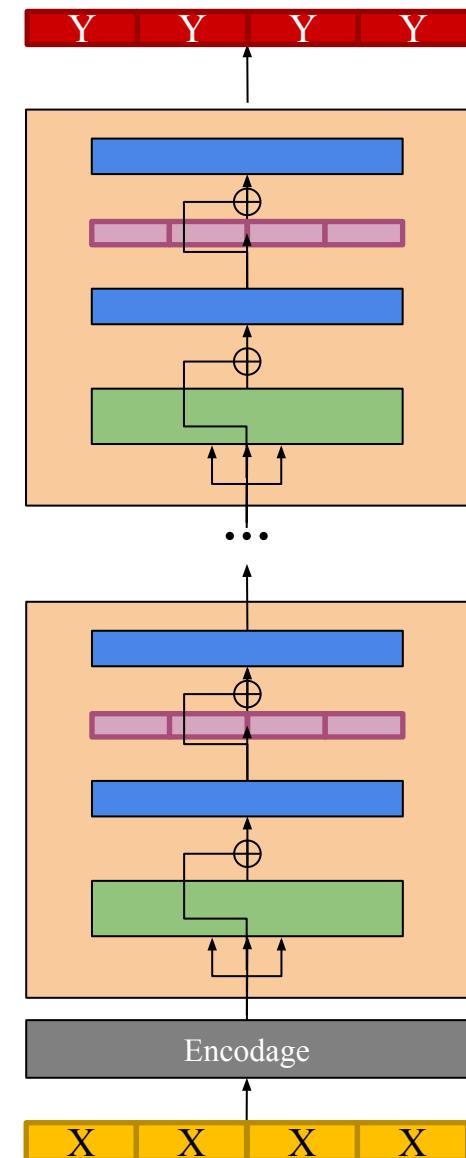
Figure 1. Trend of state-of-the-art NLP model sizes with time.

<https://developer.nvidia.com/blog/scaling-language-model-training-to-a-trillion-parameters-using-megatron/>

“355 years on a V100 GPU server with 28 TFLOPS capacity and would cost **\$4.6 million at \$1.5 per hour**”

<https://bdtechtalks.com/2020/09/21/gpt-3-economy-business-model/>

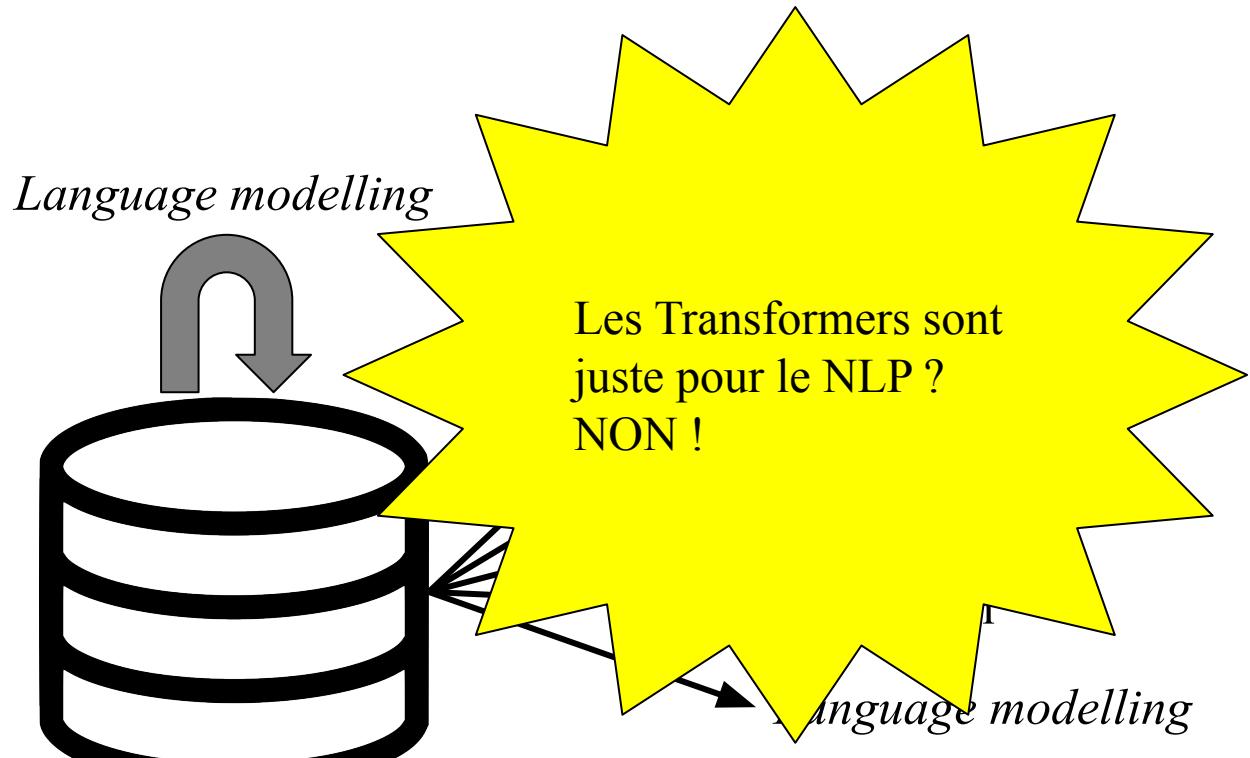
GPT1-2-3



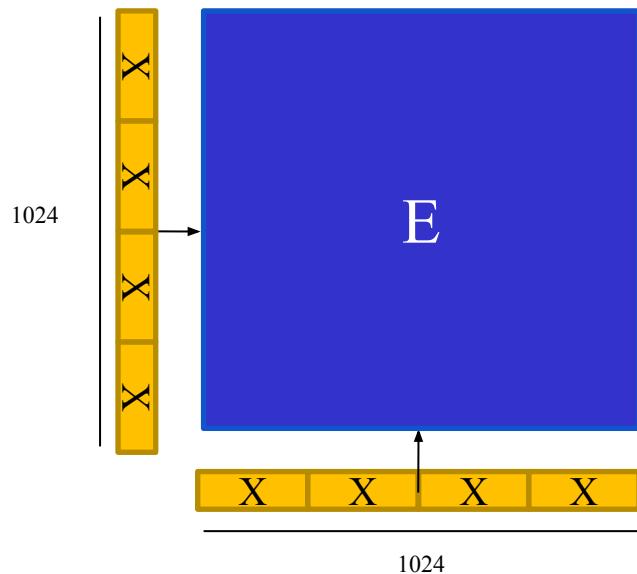
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

“Juste” l'encodeur du Transformer

Préentraîné sur un large corpus de texte, puis
finetuned sur une tâche

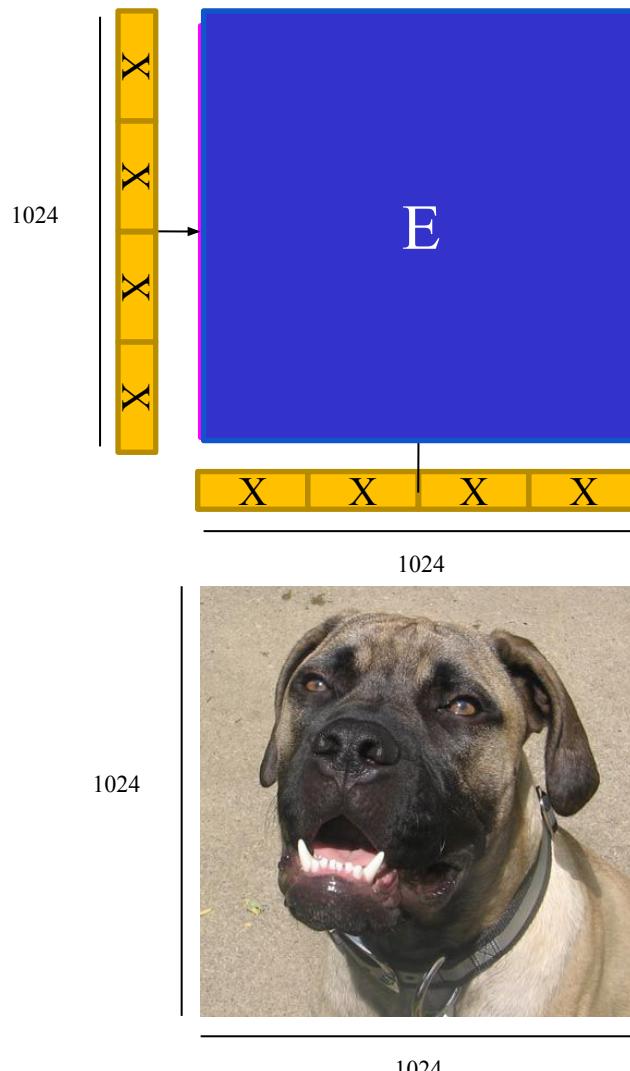


Vision Transformers (ViT)



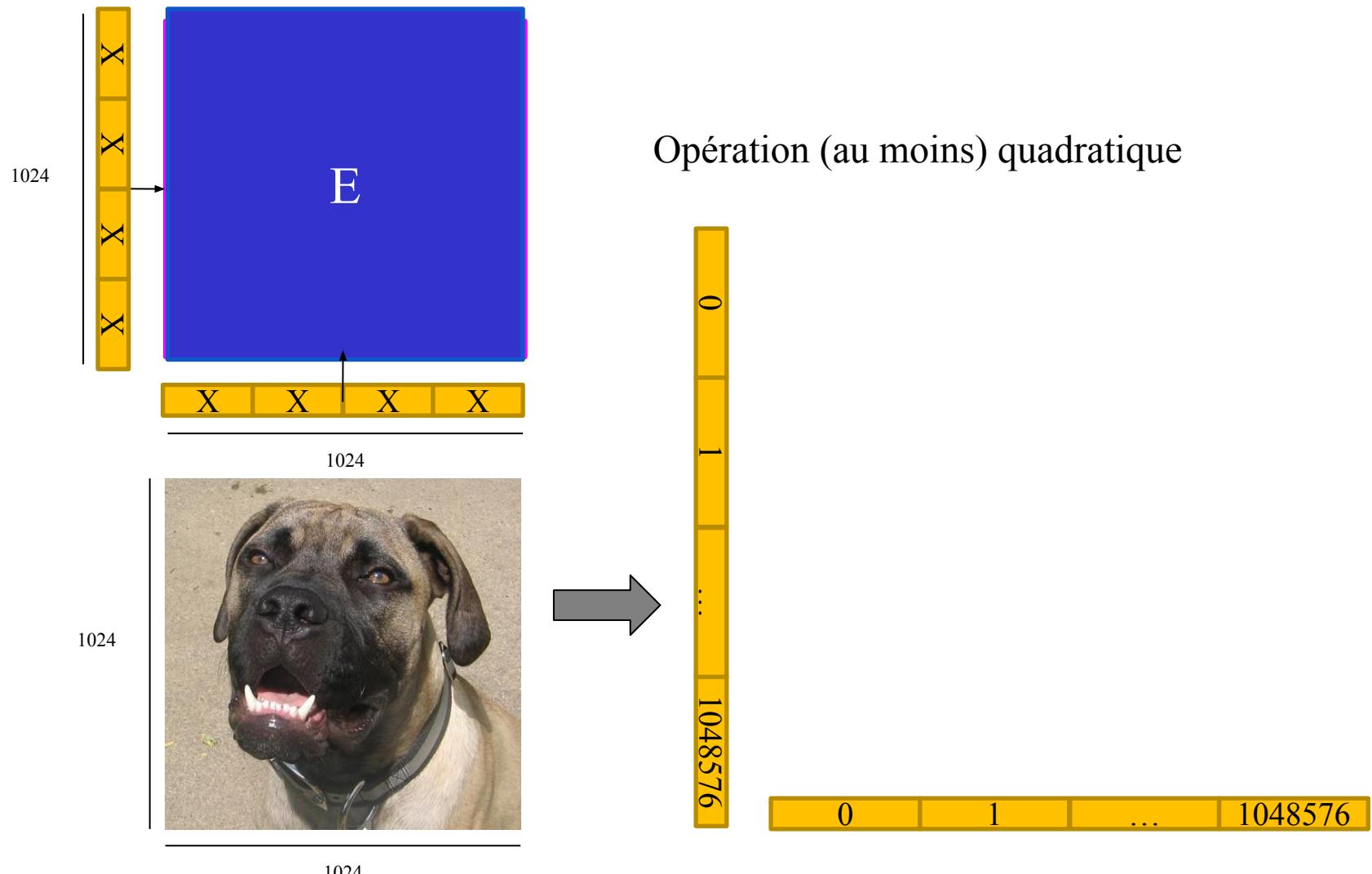
Opération (au moins) quadratique =
1024² opérations et composantes à
garder en mémoire

Vision Transformers (ViT)



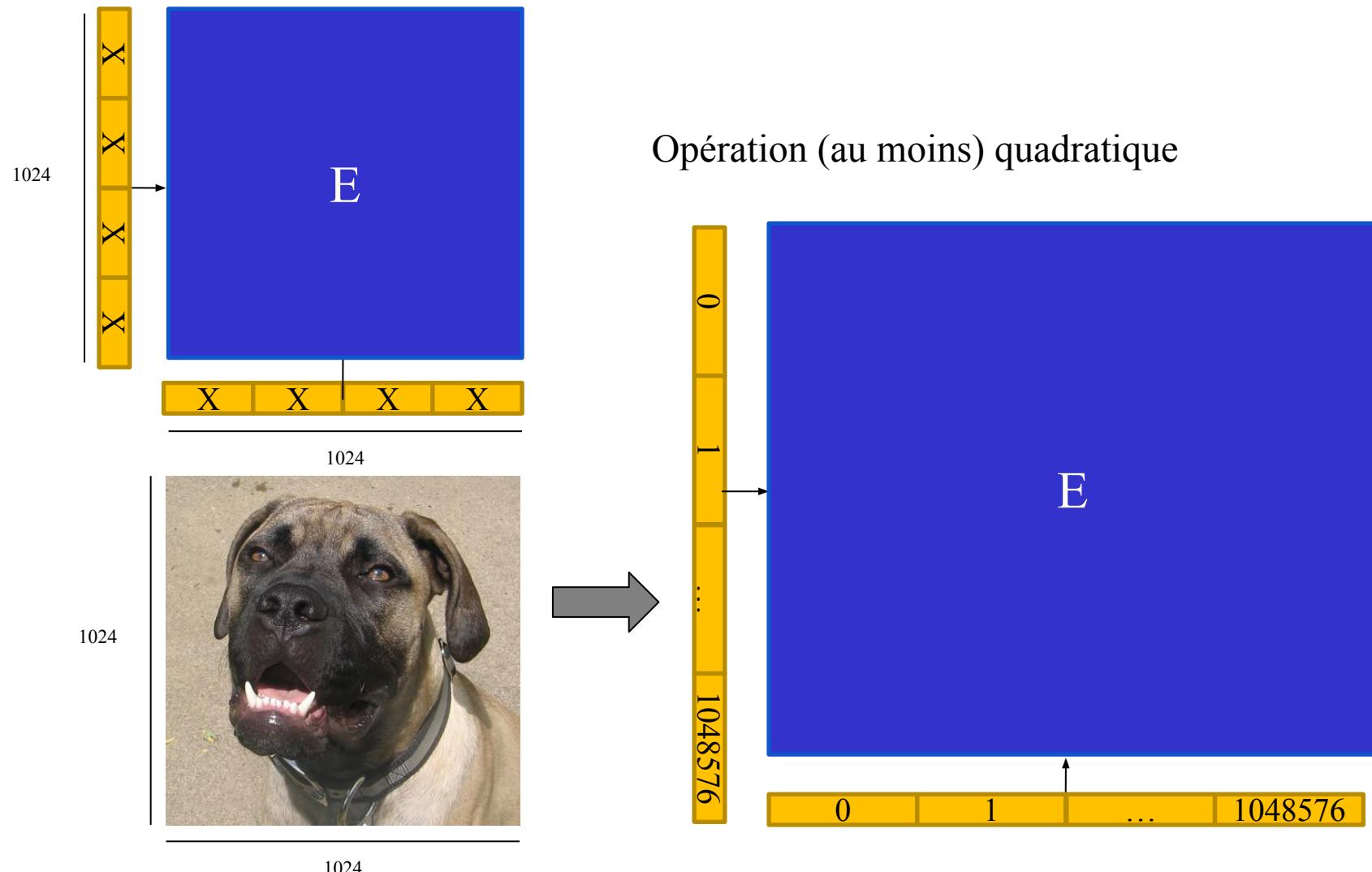
Opération (au moins) quadratique

Vision Transformers (ViT)



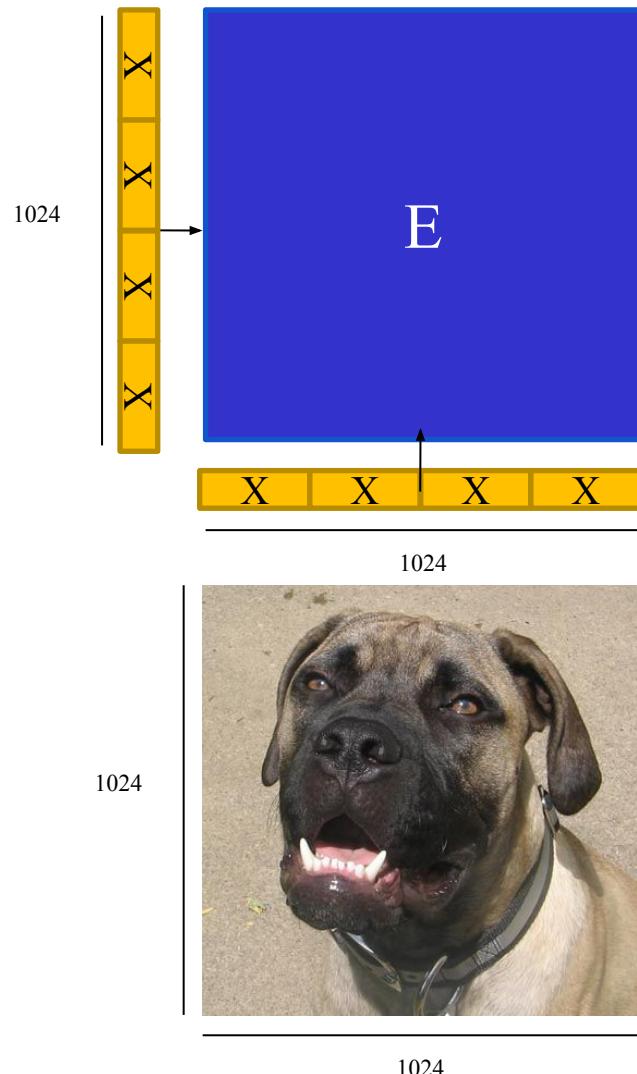
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Vision Transformers (ViT)

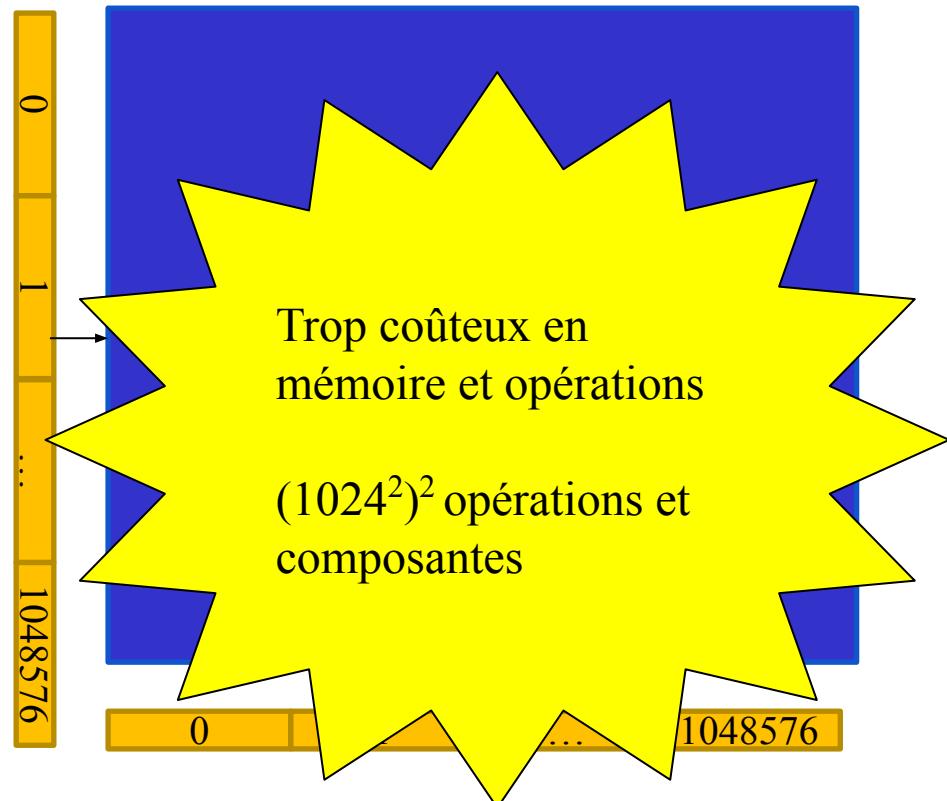


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

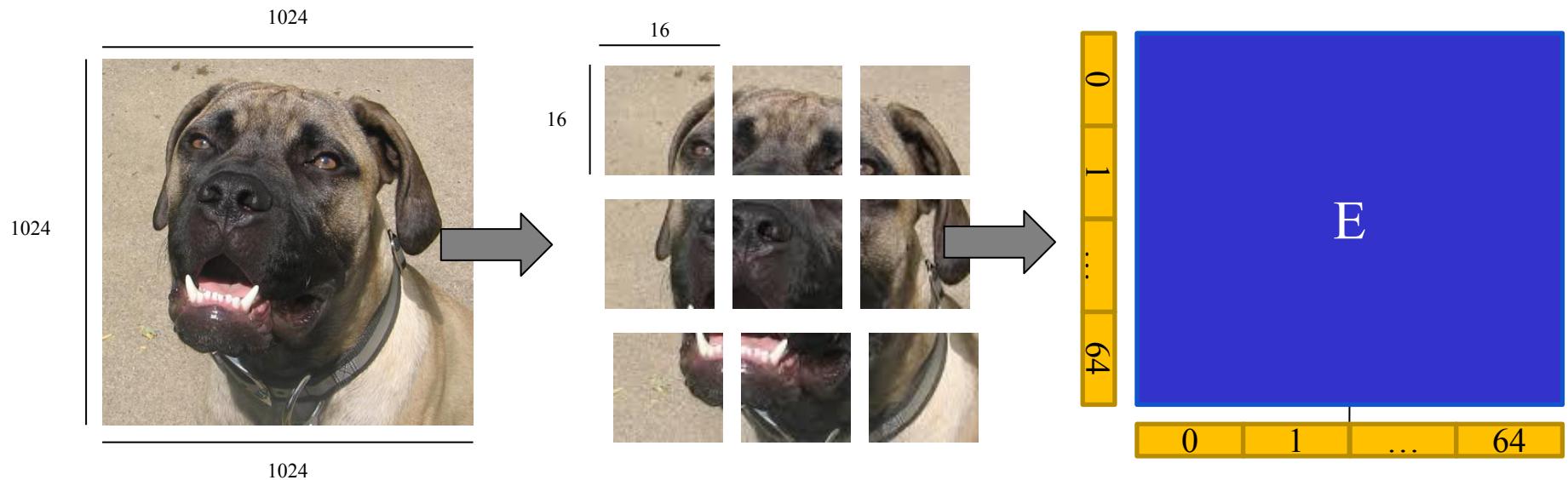
Vision Transformers (ViT)



Opération (au moins) quadratique



Vision Transformers (ViT)



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.

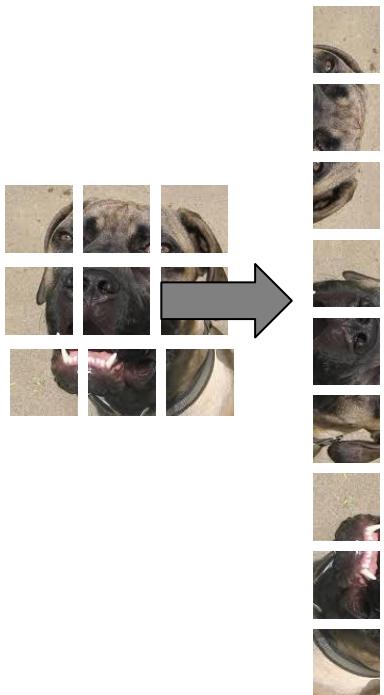
Vision Transformers (ViT)

- L'image est séparée en patch

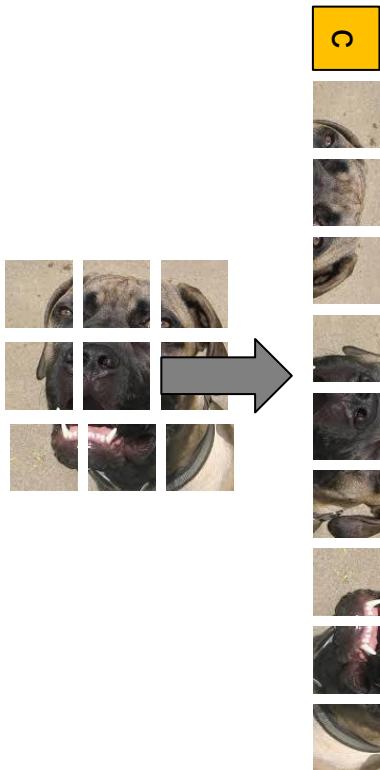


Vision Transformers (ViT)

- L'image est séparée en patch
- Chaque patch est linéarisée

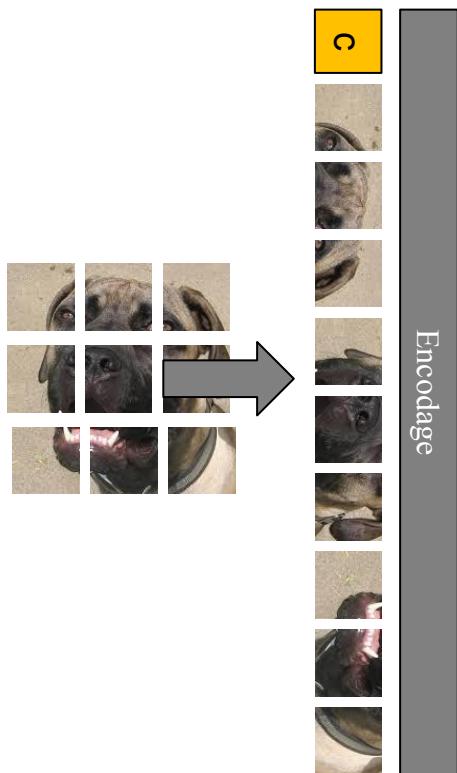


Vision Transformers (ViT)



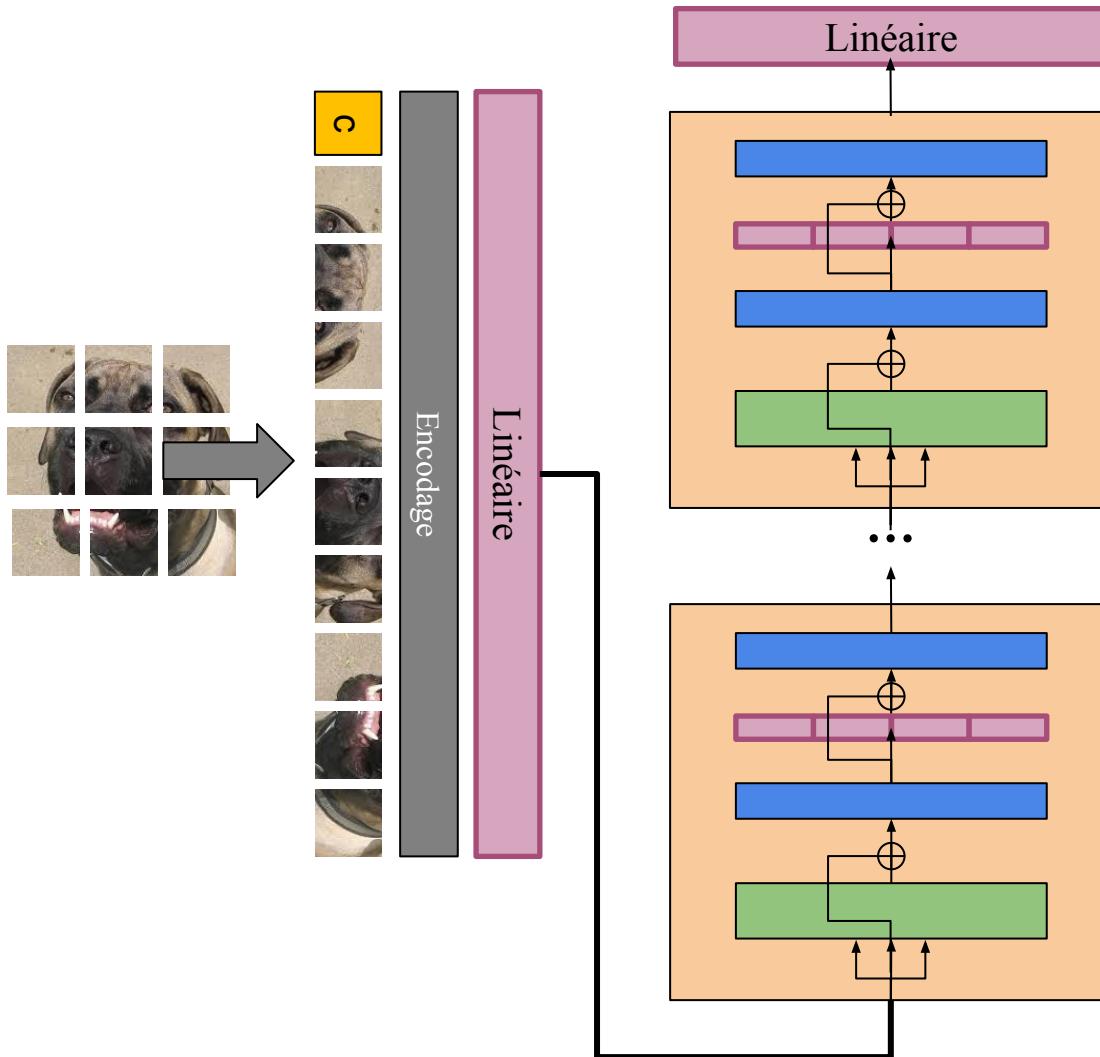
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté

Vision Transformers (ViT)



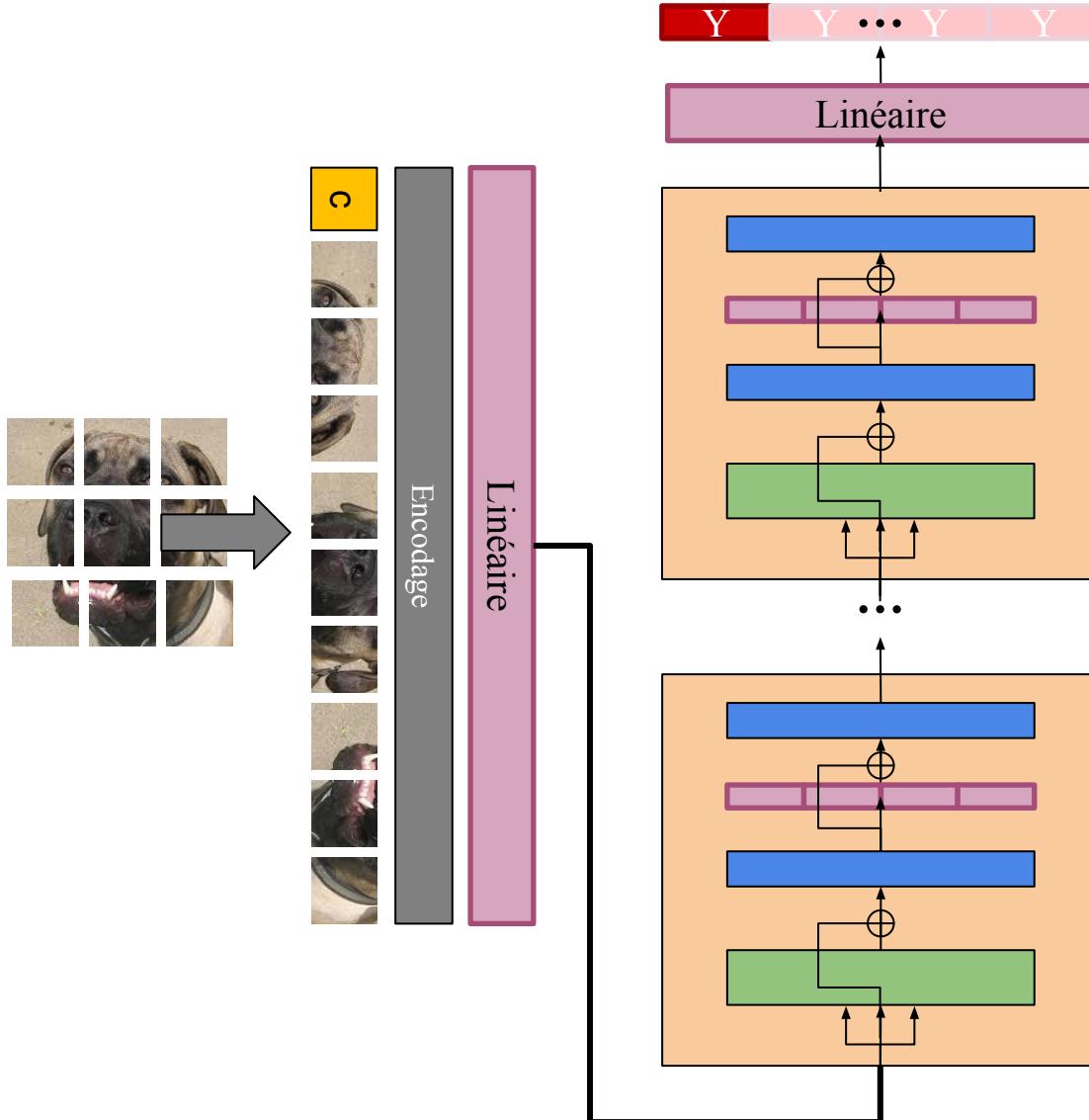
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens

Vision Transformers (ViT)



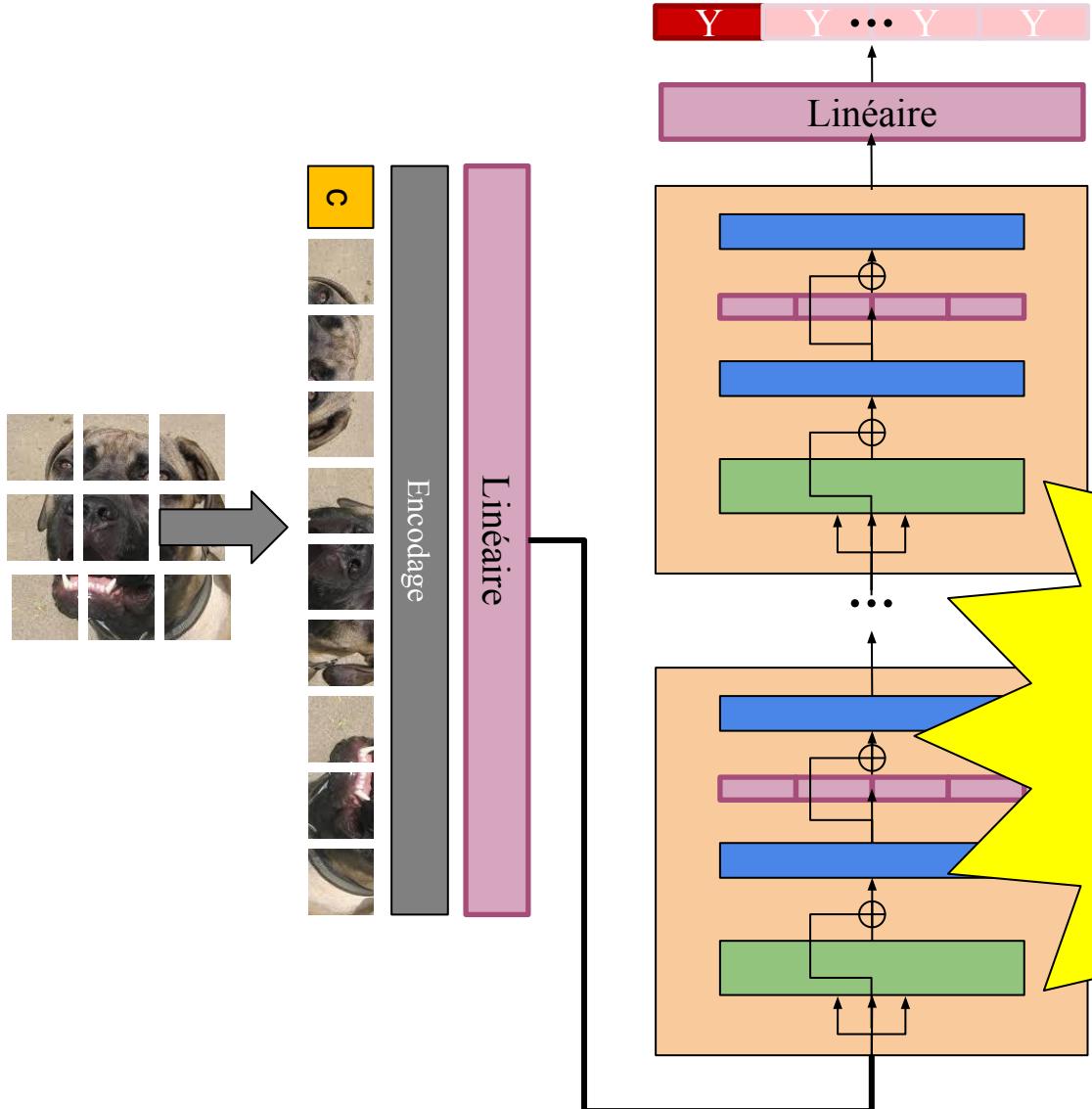
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all you need*

Vision Transformers (ViT)



- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all you need*
- Seulement la prédiction correspondant au token de classe est faite

Vision Transformers (ViT)



- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all we need*

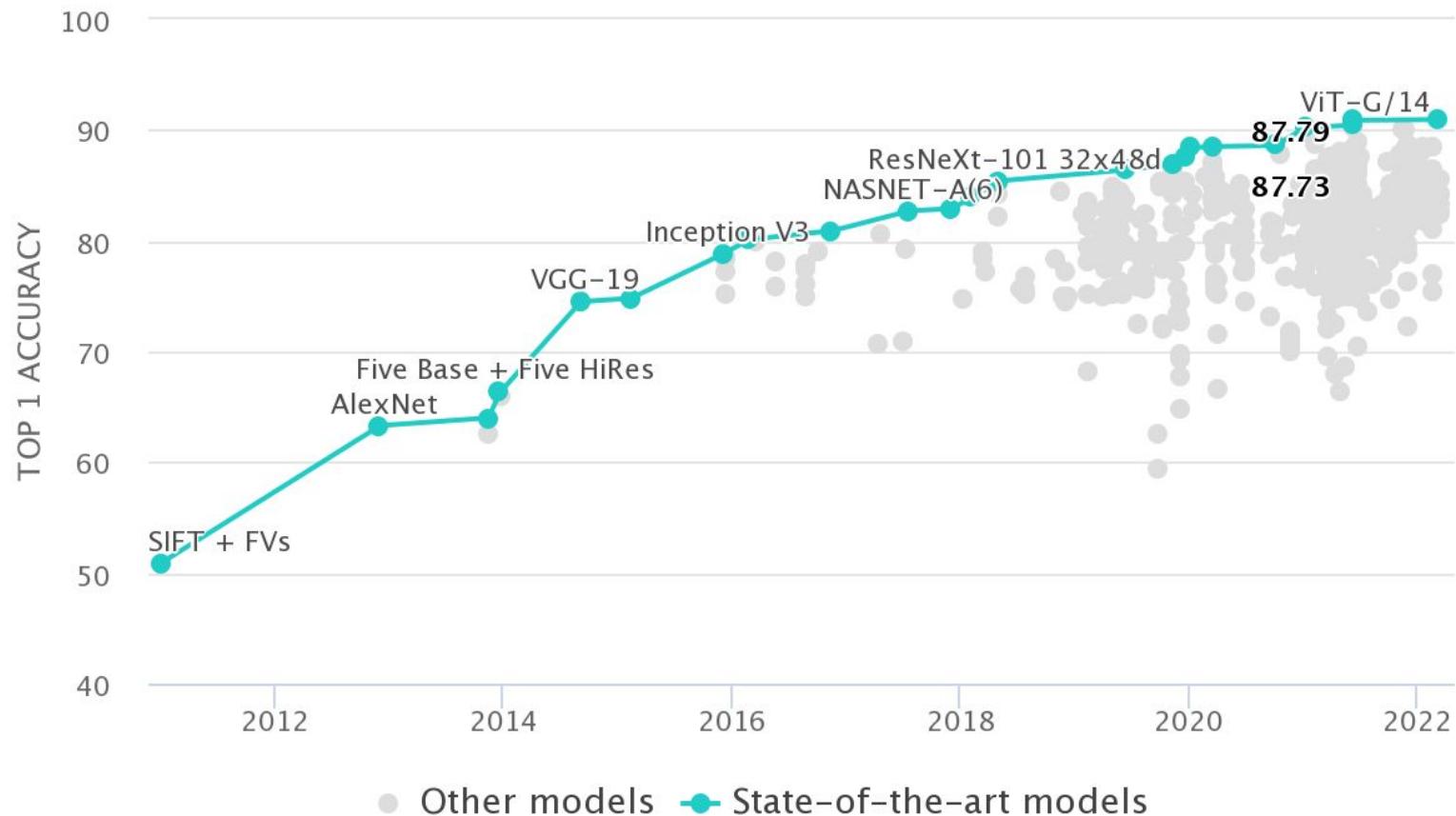
Aucune convolution !

Vision Transformers (ViT)

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

Vision Transformers (ViT)



Les *vision transformers* dominent la classification depuis leur arrivée

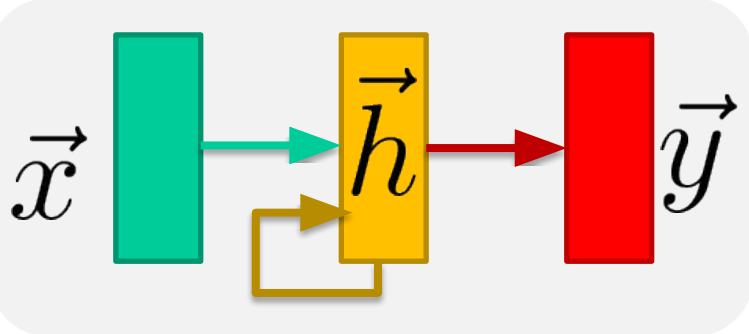
Vision Transformers (ViT)

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	V-MoE-15B (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts			2021	Transformer
6	Meta Pseudo Labels (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels			2021	EfficientNet JFT-300M
7	SwinV2-G	90.17%			✓	Swin Transformer V2: Scaling Up Capacity and Resolution			2021	Transformer
8	Florence-CoSwin-H	90.05%	99.02%		✓	Florence: A New Foundation Model for Computer Vision			2021	Transformer
9	Meta Pseudo Labels (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels			2021	EfficientNet JFT-300M
10	NFNet-F4+	89.2%		527M	✓	High-Performance Large-Scale Image Recognition Without Normalization			2021	CNN JFT-300M

Les vision transformers dominent la classification depuis leur arrivée

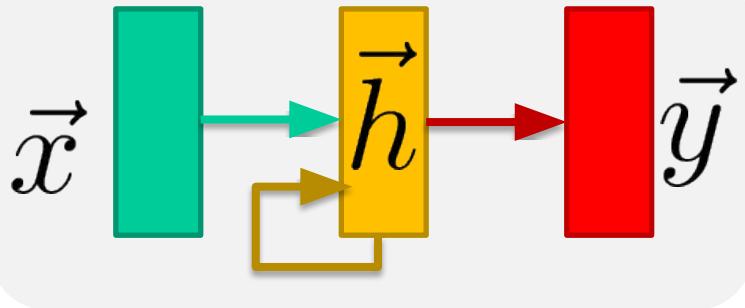
<https://paperswithcode.com/sota/image-classification-on-imagenet>

Sommaire

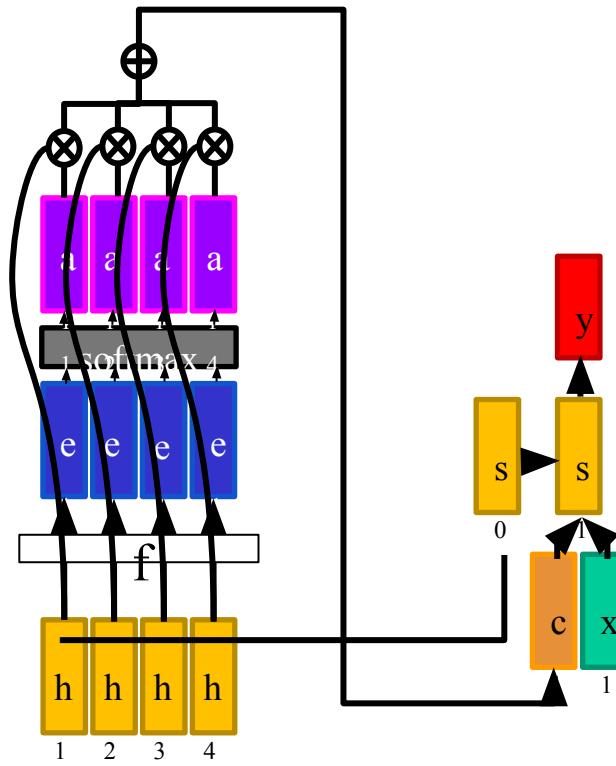


- Les réseaux récurrents peuvent traiter des séquences
- Ils ne requièrent que de légères modifications à des réseaux pleinement connectés
- Ils sont instables sur de longues séquences
- LSTM/GRU sont utilisés en pratique

Sommaire



- Les réseaux récurrents peuvent traiter des séquences
- Ils ne requièrent que de légères modifications à des réseaux pleinement connectés
- Ils sont instables sur de longues séquences
- LSTM/GRU sont utilisés en pratique



- L'attention est un mécanisme très puissant permettant aux réseaux d'apprendre quelle partie des données utiliser pour faire une prédiction
- L'attention n'est pas limité au texte, ou même aux séquences

Sommaire

- Les *Transformers* sont un modèle extrêmement puissant pour toutes tâches reliées au *langage naturel*
- Les transformers n'utilisent *que* l'attention
- Les transformers sont demandant en ressources
- Ceux-ci ne sont pas limités au langage !

