

Réseaux de neurones

IFT 780

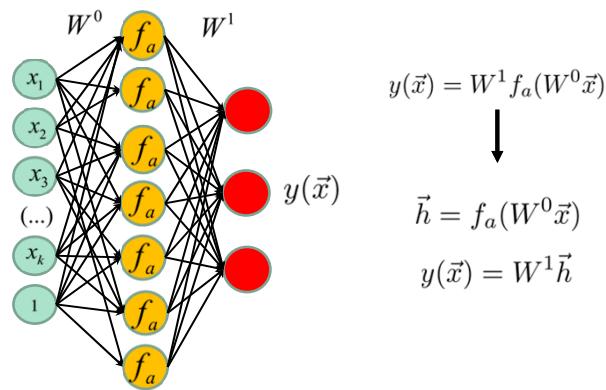
Réseaux récurrents

Par

Pierre-Marc Jodoin, Antoine Théberge

1

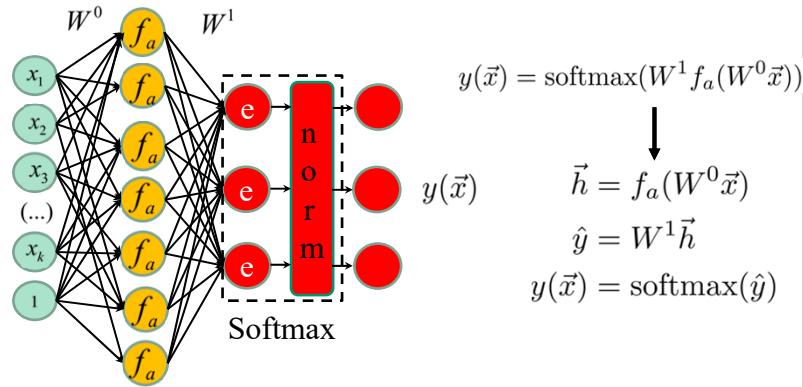
Réseau de neurones de base (régression)



f_a : fonction d'activation

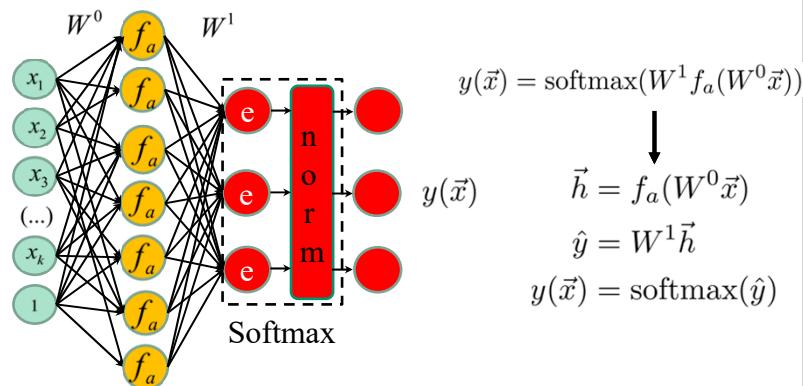
2

Réseau de neurones de base (classification)



3

Réseau de neurones de base (classification)

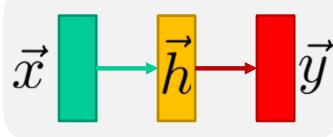


Ne permet que des tâches “1 pour 1”

- Classification (1 image = 1 étiquette)
- Régression (1 donnée = 1 vecteur)
- Localisation (1 boîte = 1 classification + 1 régression)

4

Illustration simplifiée

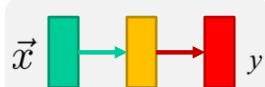


RN de base

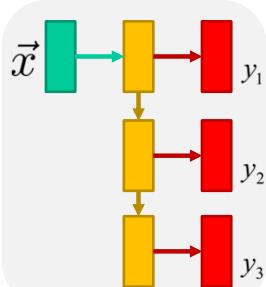
5

Différentes configurations pour différentes applications

1 entrée et 1 sortie

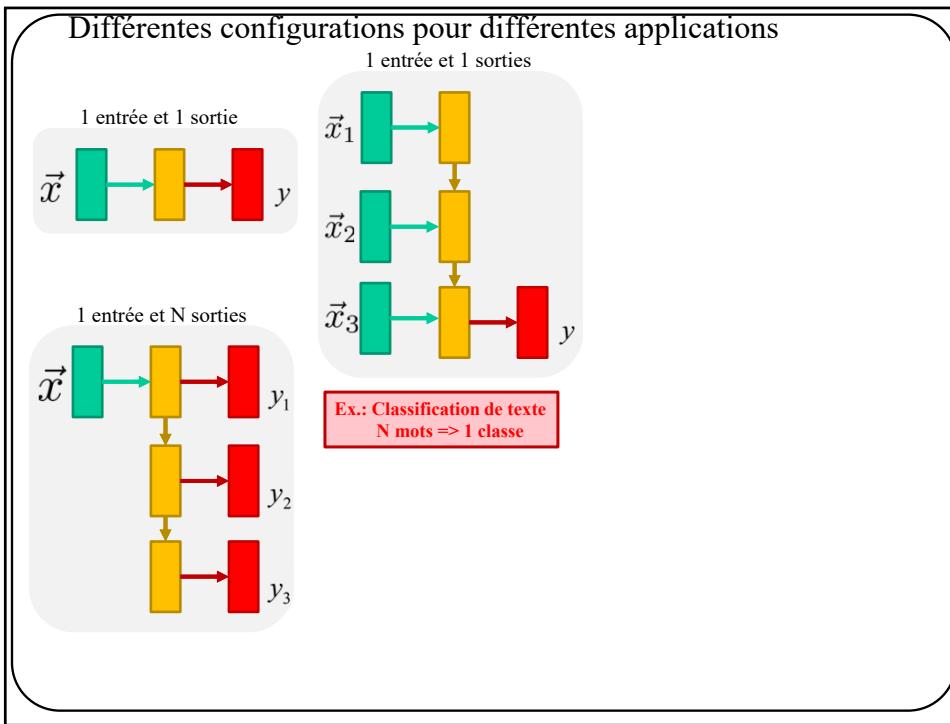


1 entrée et N sorties

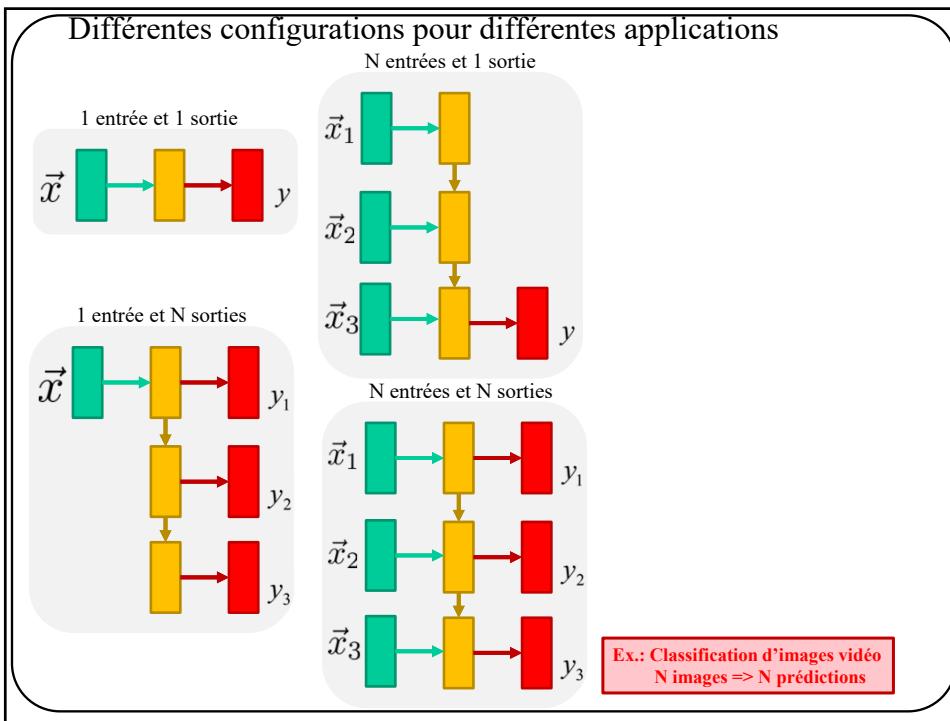


**Ex.: description d'une image
1 image => N mots**

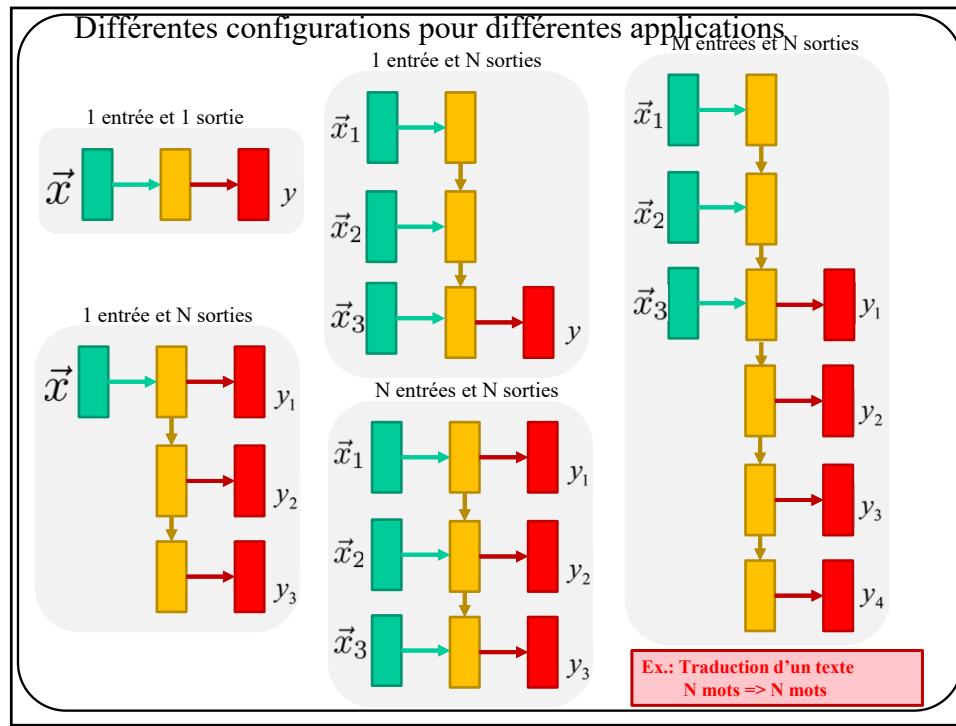
6



7

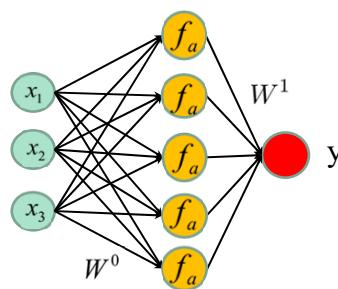


8



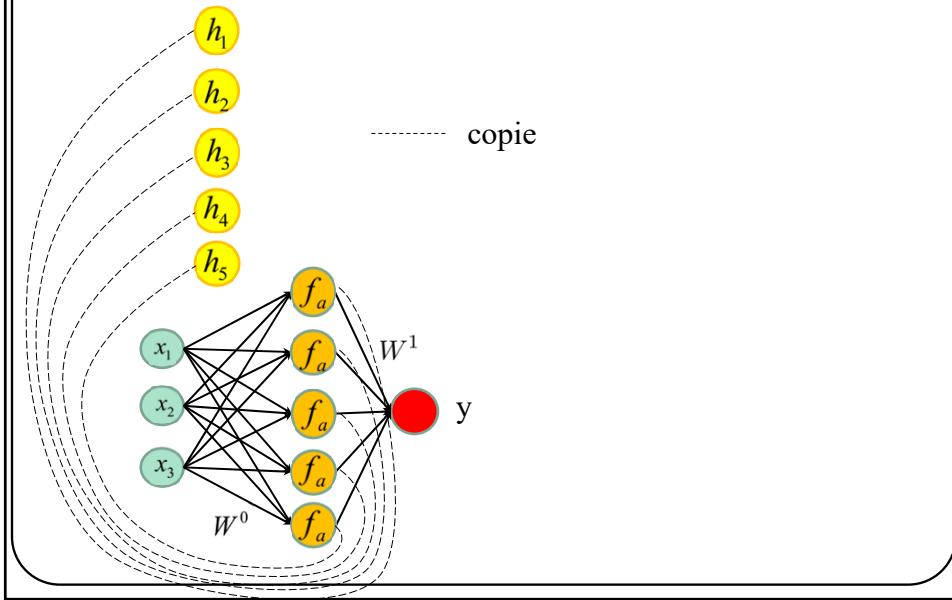
9

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



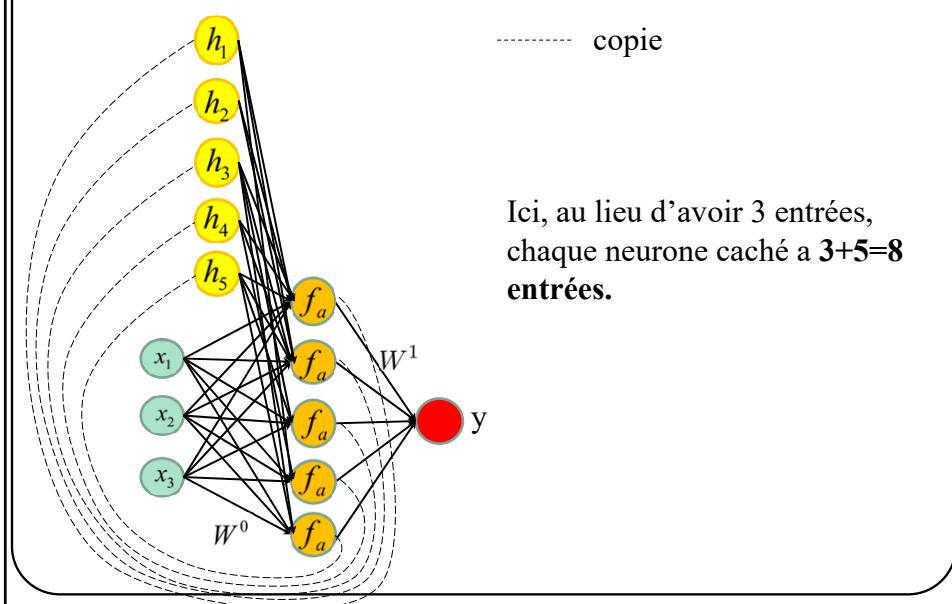
10

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



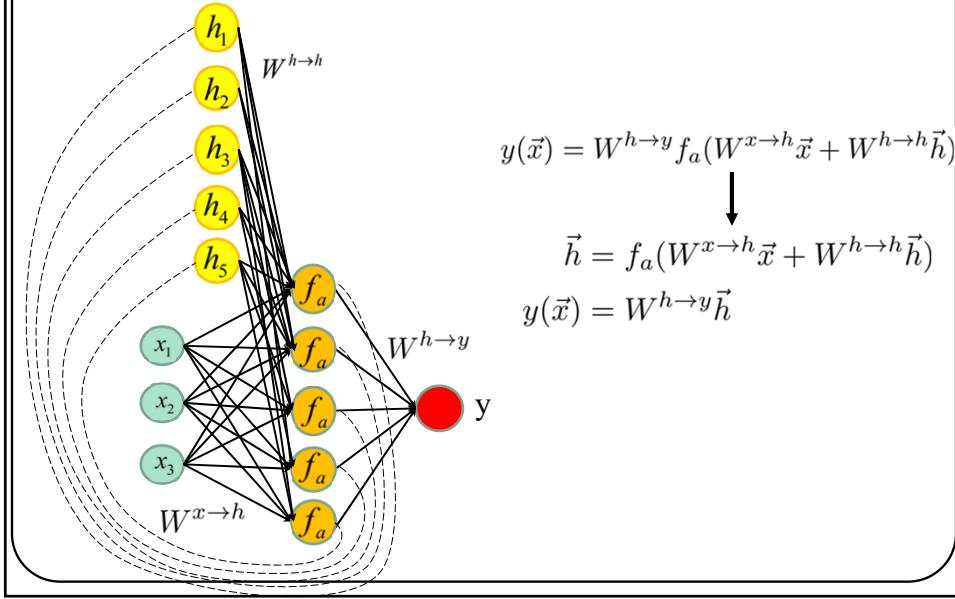
11

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



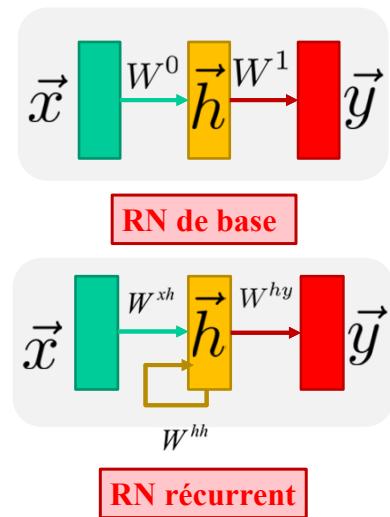
12

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



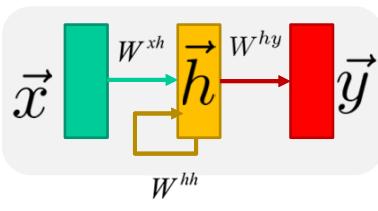
13

Illustration simplifiée



14

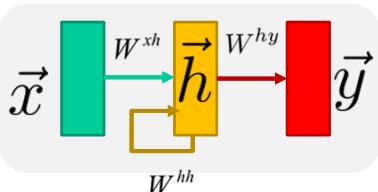
Dans le cas général avec K sorties (régression)



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

15

Dans le cas général avec K sorties (classification)

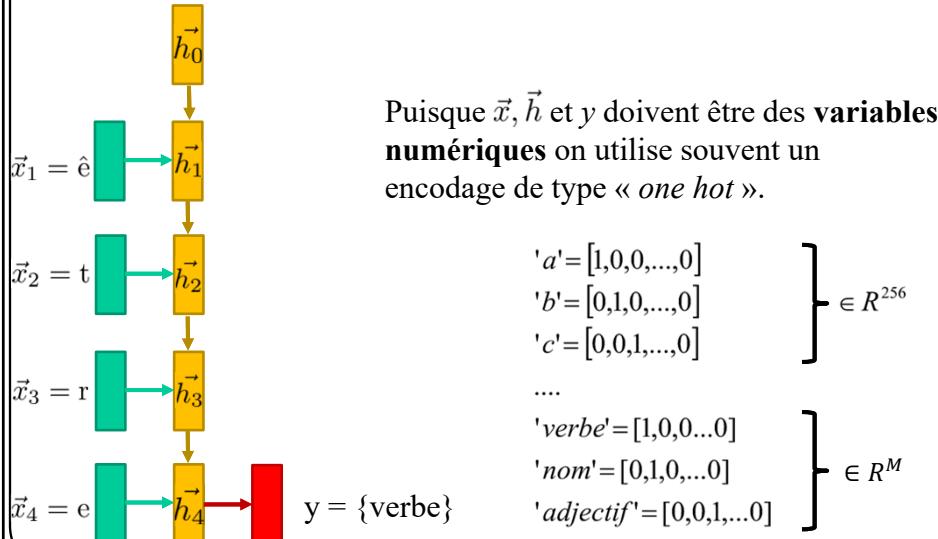


$$\begin{aligned} \vec{y}(\vec{x}) &= W^{hy} f_a(W^{xh} \vec{x} + W^{hh} \vec{h}) \\ \vec{h} &= f_a(W^{xh} \vec{x} + W^{hh} \vec{h}) \\ \hat{y} &= W^{hy} \vec{h} \\ \vec{y}(\vec{x}) &= \text{softmax}(\hat{y}) \end{aligned}$$

16

Exemple pour N entrées et 1 sortie:

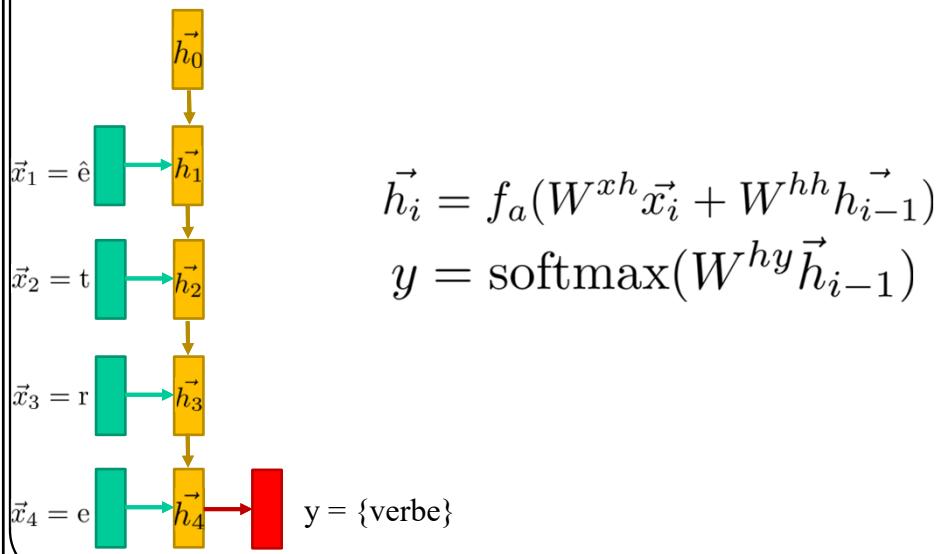
Analyse grammaticale (classification) : (\hat{e}, t, r, e) $\Rightarrow \{\text{verbe}\}$



17

Exemple pour N entrées et 1 sortie:

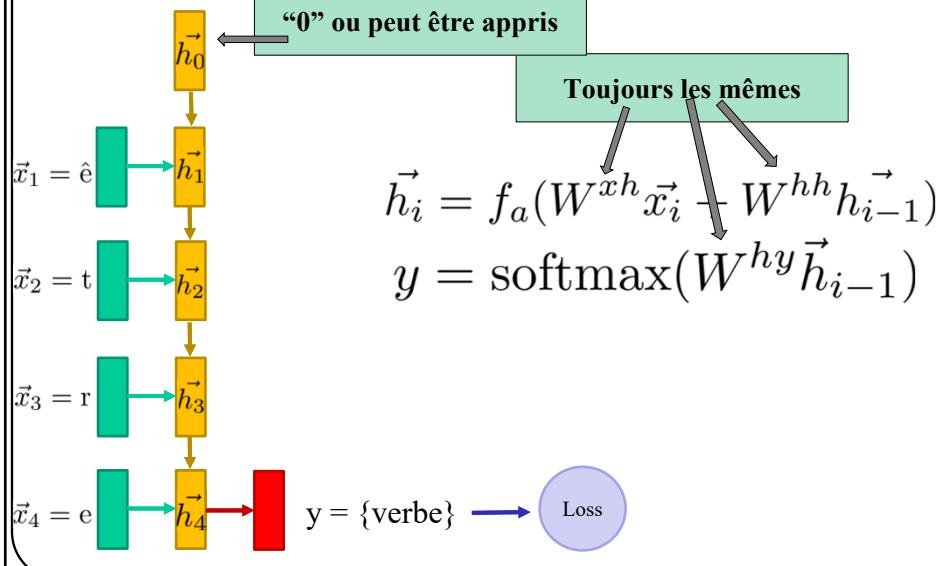
Analyse grammaticale (classification) : (\hat{e}, t, r, e) $\Rightarrow \{\text{verbe}\}$



18

Exemple pour N entrées et 1 sortie:

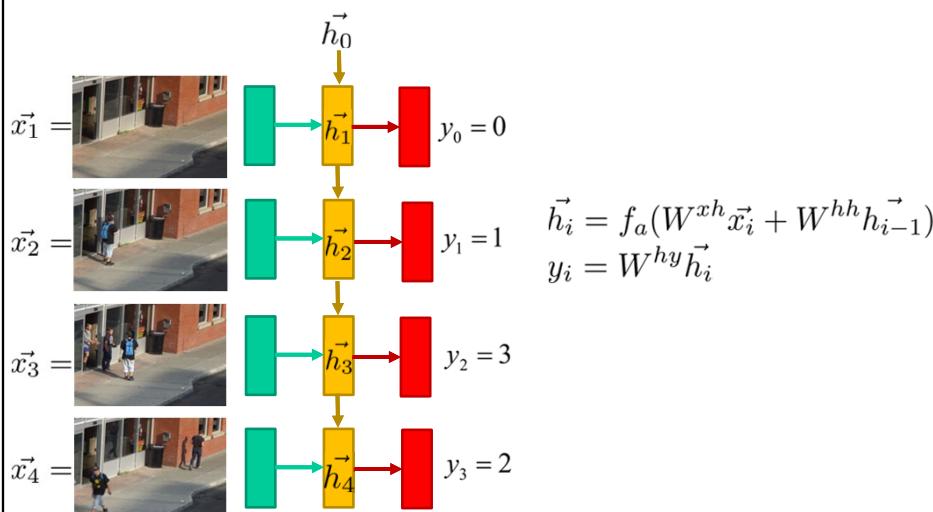
Analyse grammaticale (classification) : (é.t.r.e)=> {verbe}



19

Même idée pour N entrées et N sorties:

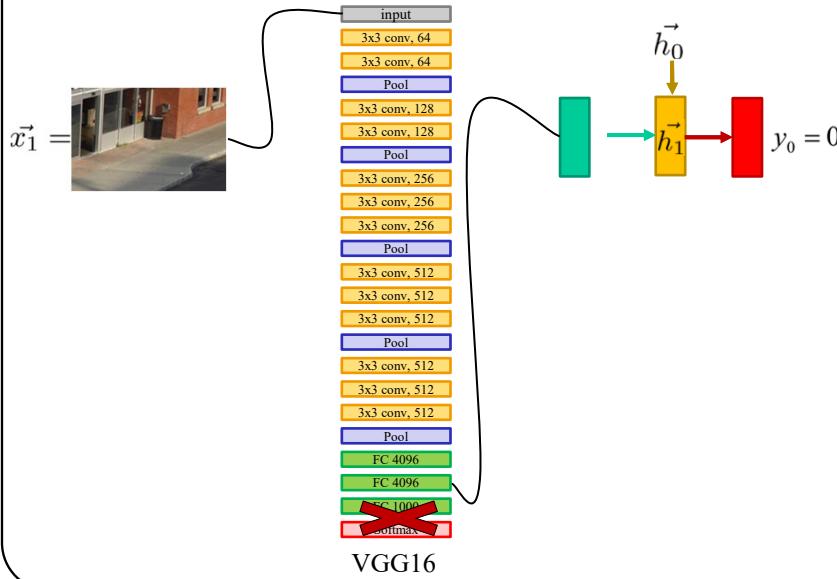
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



20

Même idée pour N entrées et N sorties:

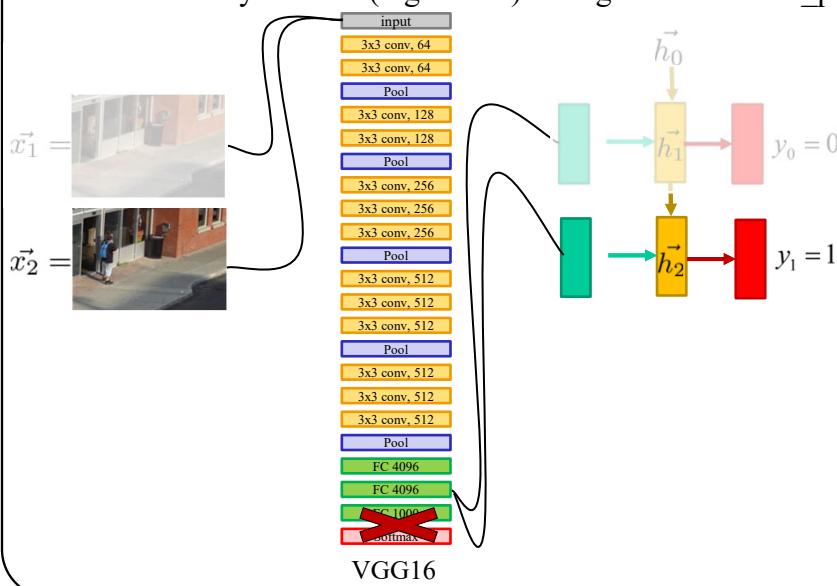
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



21

Même idée pour N entrées et N sorties:

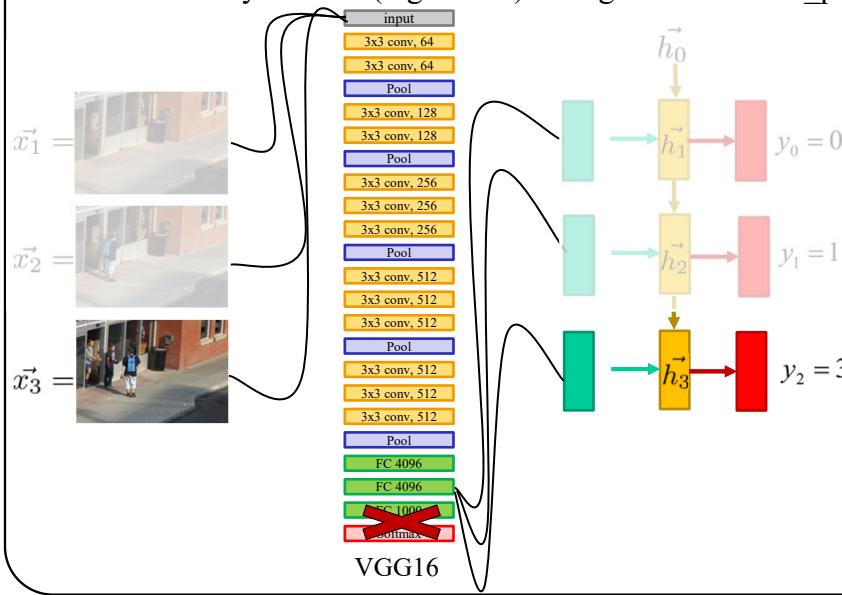
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



22

Même idée pour N entrées et N sorties:

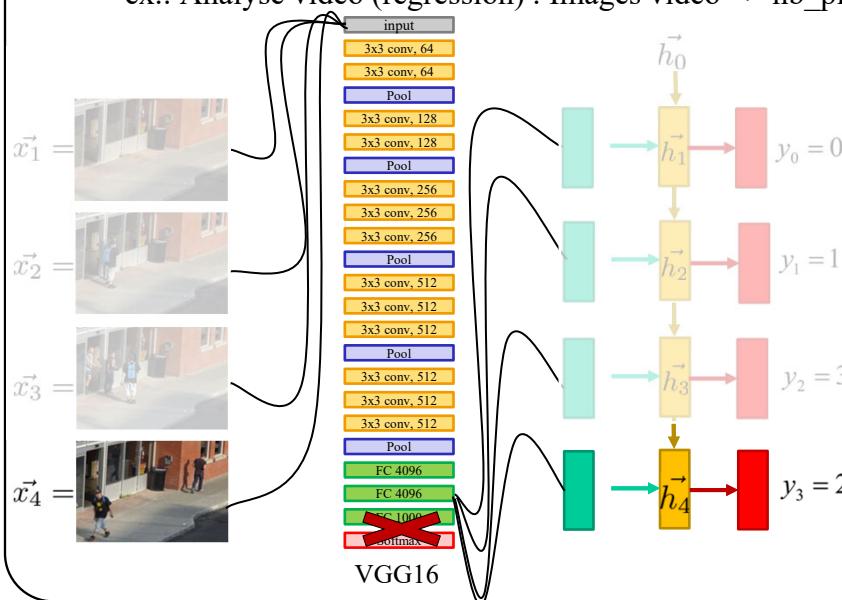
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



23

Même idée pour N entrées et N sorties:

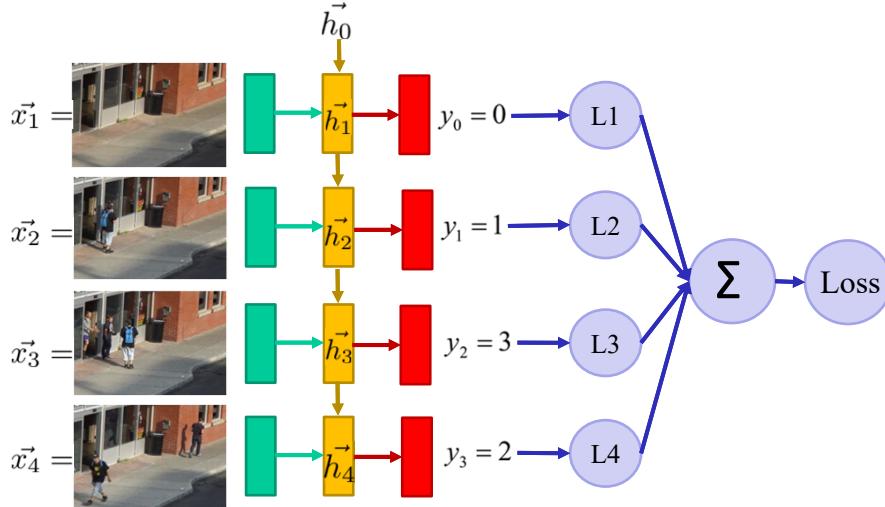
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



24

Même idée pour N entrées et N sorties:

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



25

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

26

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

$$\vec{x}_1 = m \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \rightarrow$$

27

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

$$\vec{x}_1 = m \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{array}{c} W^{xh} \\ \downarrow \\ W^{hh} \end{array}$$

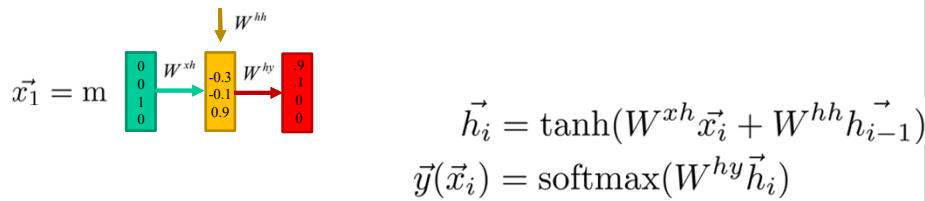
$$\vec{h}_i = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$

28

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

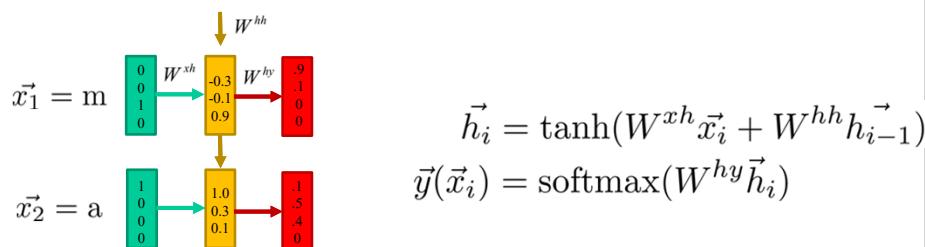


29

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

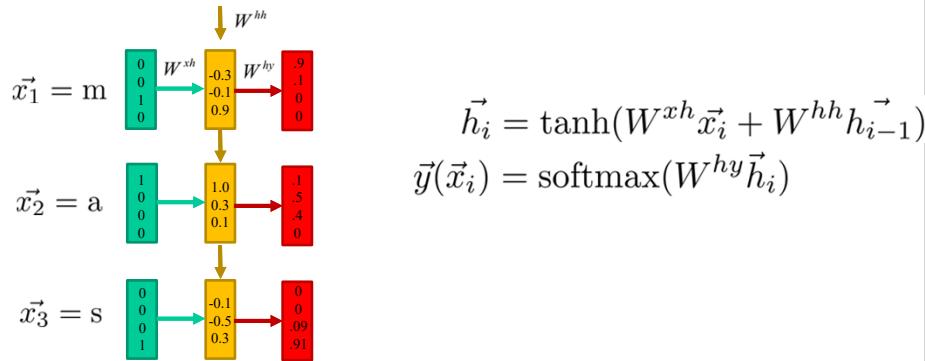


30

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

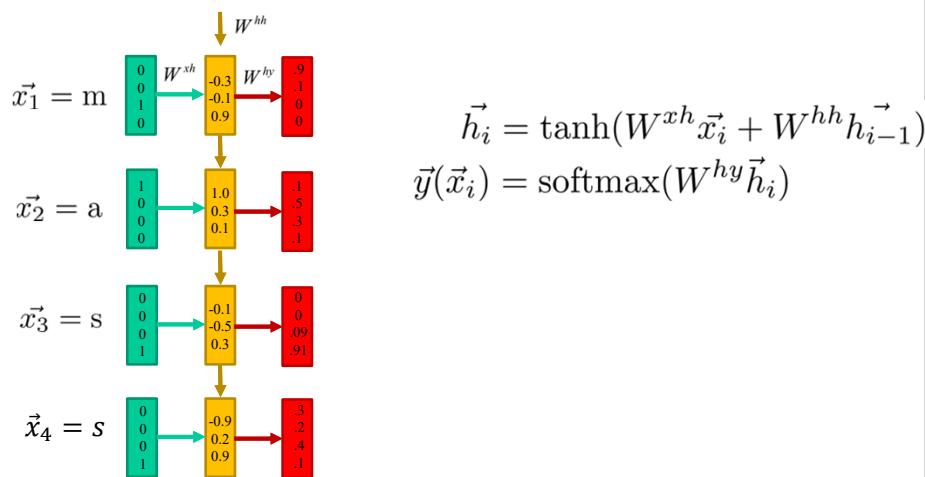


31

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

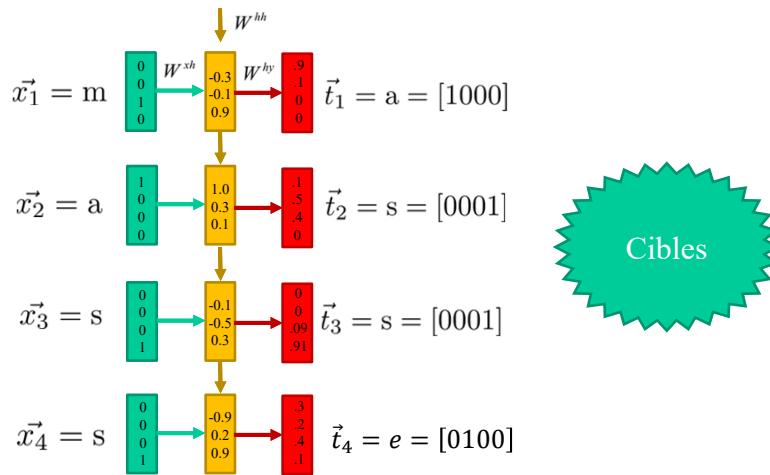


32

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

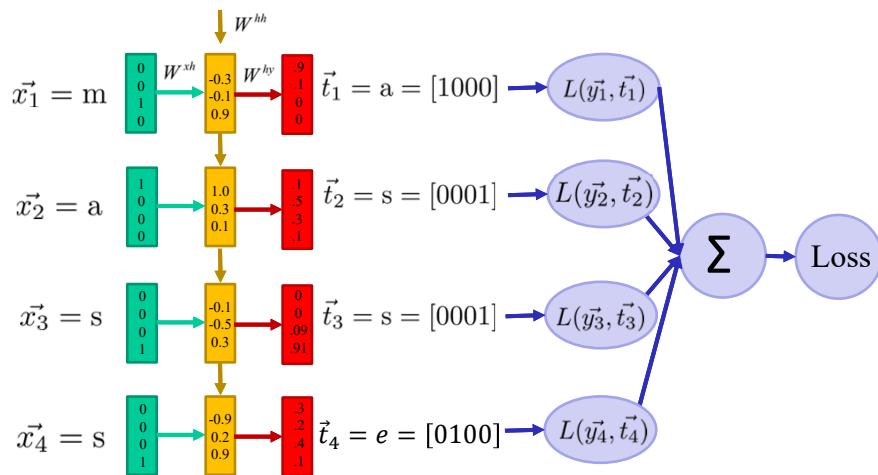


33

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

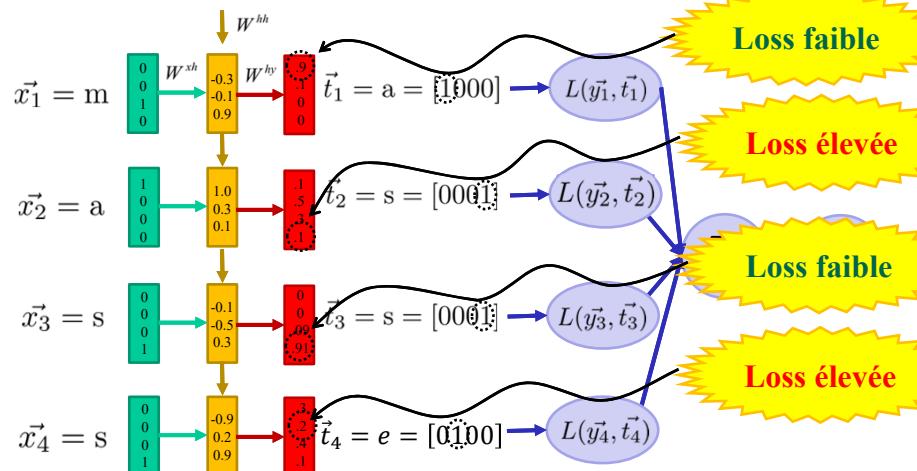


34

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

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

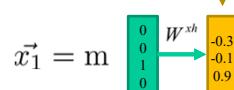


35

Autre exemple: prédiction de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



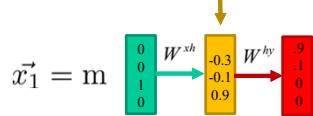
Étape 1 : Calcul de la couche cachée

36

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



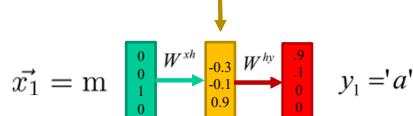
Étape 2 : Calcul de la sortie (softmax)

37

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



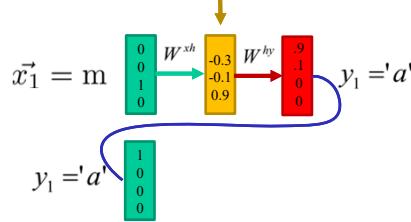
Étape 3 : Sélectionner le caractère le plus probable

38

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



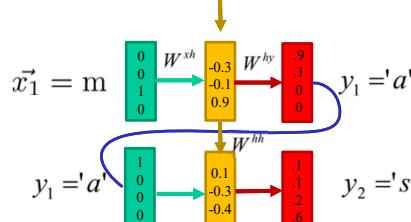
Étape 4 : Injecter le caractère prédit au début du réseau

39

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



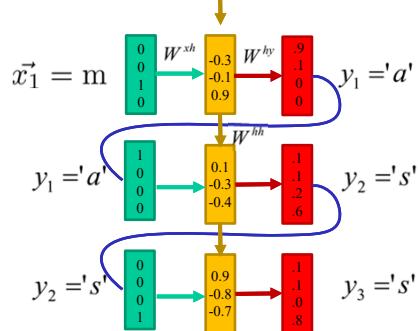
Et on recommence!

40

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres

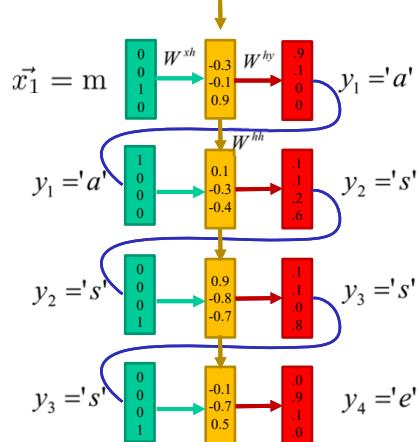


41

Autre exemple: prédition de caractères (modèle de langue)

Alphabet : [a,e,m,s]

En test : prédire les lettres les unes après les autres



42

Autre exemple: prédition 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 'there are %s unique characters, %s 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 Wih = 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 = list(inputs), list(targets)
30     hprev = np.array([hprev])
31     loss = 0
32     npx = np.zeros((vocab_size, hidden_size))
33     npnx = np.zeros((hidden_size, vocab_size))
34     npnx[0] = hprev
35     npnx[1] = np.zeros((hidden_size, 1))
36     npnx[2] = np.zeros((vocab_size, 1))
37     npnx[3] = np.zeros((hidden_size, 1))
38
39     for i in range(len(inputs)):
40         x = inputs[i]
41         xenc = np.zeros((vocab_size, 1))
42         xenc[0][int(x)] = 1
43
44         npnx[0] = hprev
45         npnx[1] = npnx[2]
46         npnx[2] = npnx[3]
47         npnx[3] = npnx[1]
48
49         npnx[1] = np.tanh(npx[0].dot(Wih)+xenc.dot(Whh)+bh)
50         npnx[2] = np.tanh(npx[1].dot(Whh)+bh)
51         npnx[3] = np.tanh(npx[2].dot(Why)+by)
52
53         loss += -np.log(npx[3][int(targets[i])])
54
55         npx[0] = npnx[0]
56         npx[1] = npnx[1]
57         npx[2] = npnx[2]
58         npx[3] = npnx[3]
59
60     return loss, npnx[3]
```

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

...

43

Autre exemple: prédition 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,
But as the riper should by time decay,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-delusional fuel,
Making thy suns by thine own hands to be
Thyself thy foe, to thy sweet self too cruel:
Thou art now the world's fresh ornament,
And only herald to the gaudy spring.
Within thine outward seeming all is silent,
And thou that make'st woe in negliging:
Pity the world, or else this glutton be,
To use 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 gared on now,
Will闹 and garnish thy 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 thirtless praise.
How man would poor deserv'd ill thine beauty's use,
If these vander'd never? This fair child of mine
Shall sum my count, and make my old excuse,
Proving his beauty by succession thine!

Thou were to be new made when thou art old,
And see thy blood warm when thou feel it cold.

44

Autre exemple: prédition de caractères (modèle de langue)

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

↓ train more

"Tmont thithey" fomescerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl 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.

Crédit: A. Karpathy, CS231

45

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 ape, 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.

Crédit: A. Karpathy, CS231

46

Entraînement sur le code source de Linux en C++

```

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 << i))
            pipe = (in_use & UNXTHREAD_UNCCA) +
                ((count & 0x0000000fffff0) & 0x000000f) << 8;
        if (count == 0)
            sub(pipe, ppc_md.kexec_handle, 0x2000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rex_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/e20.h>
#include <asm/system_info.h>
#include <asm/seew.h>
#include <asm/pgproto.h>

#define REQ_PQ    vesa_slot_addr_pack
#define PPN_NOCOMP AFSR(0, load)
#define STACK_DDR(type)    (func)

#define SWAP_ALLOCATE(nr)    (0)
#define emulate_sigs() arch_get_unaligned_child()
#define access_rwt(wt)    asm volatile("movd %0, %1" : "+r" (0) : "r" (0));
if (_type & DO_READ)

static void stat_PCIE __read_mostly offsetof(struct seq_argqueue,
pC>[1]));

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

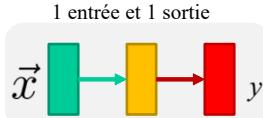
```

Crédit: A. Karpathy, CS231

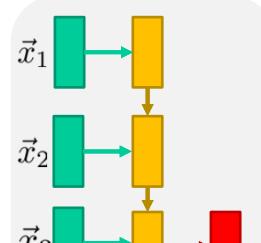
47

Différentes configurations pour différentes applications

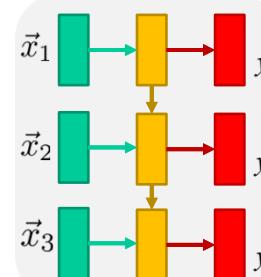
1 entrée et N sorties



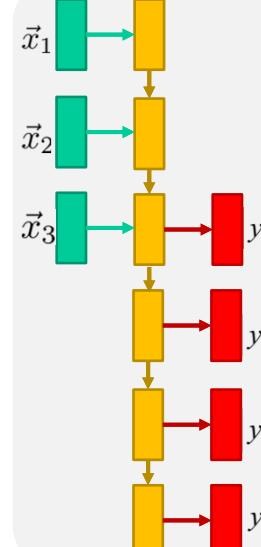
M entrées et N sorties



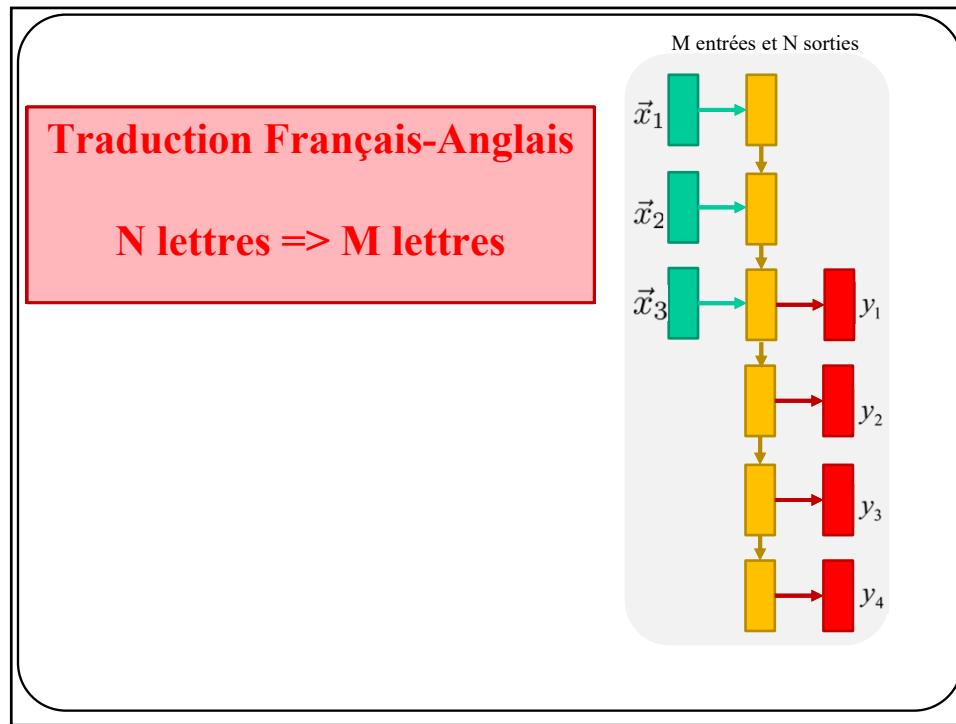
N entrées et N sorties



M entrées et N sorties



48



49

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 que de sorties !
(BoS : Begining of Sentence, EoS:End of Sentence).

50

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>]



51

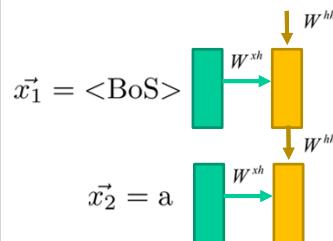
Autre exemple: traduction

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>]



52

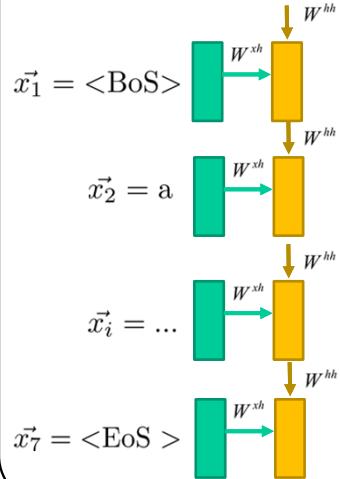
Autre exemple: traduction

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>]



53

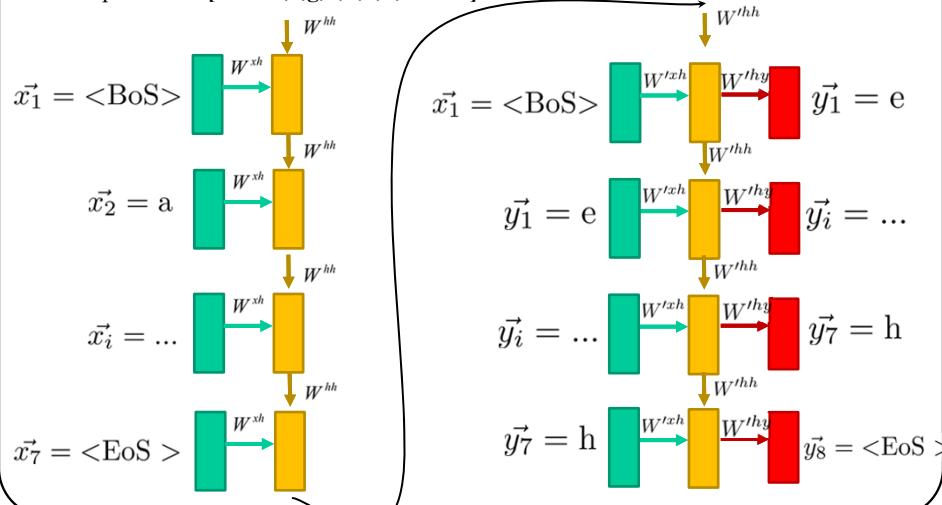
Autre exemple: traduction

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>]

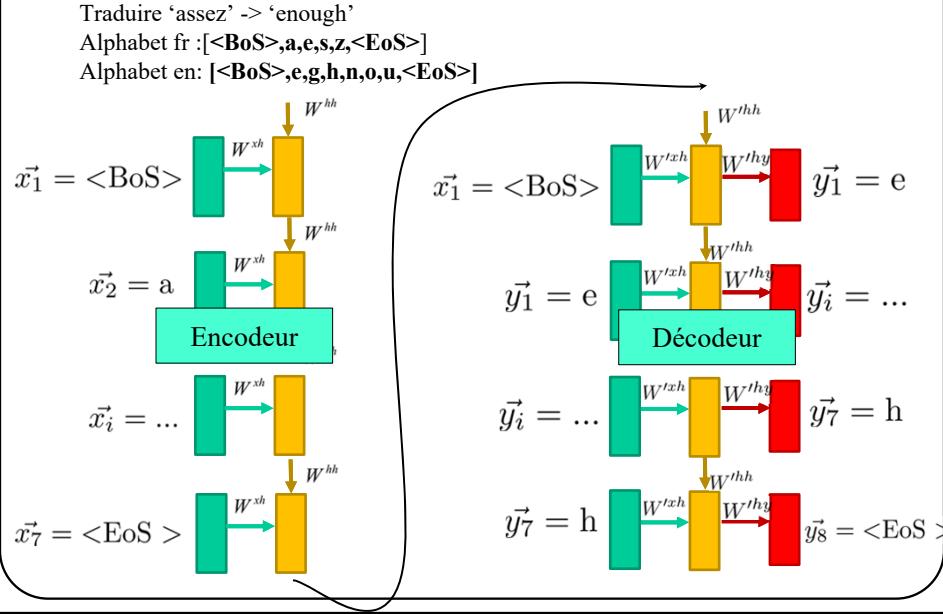
Alphabet en: [<BoS>,e,g,h,n,o,u,<EoS>]



54

Autre exemple: traduction

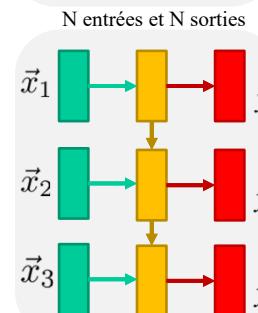
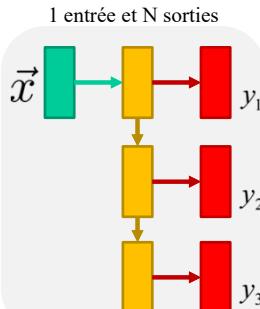
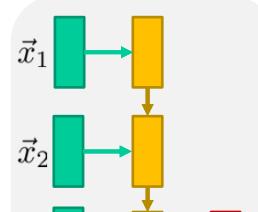
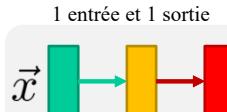
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.



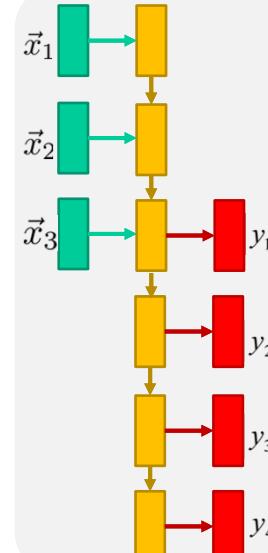
55

Différentes configurations pour différentes applications

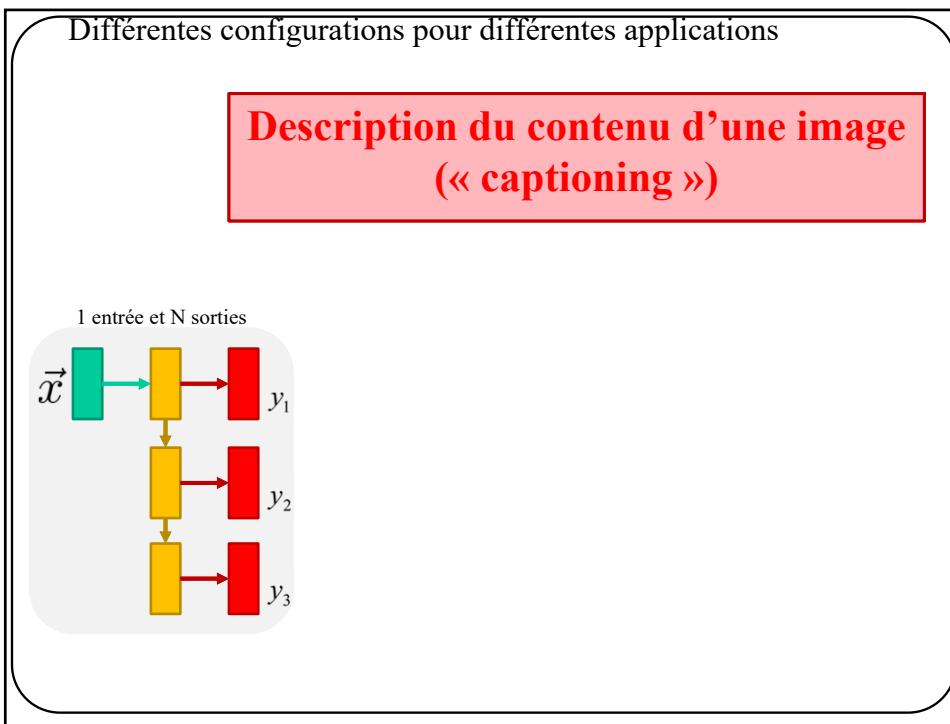
1 entrée et N sorties



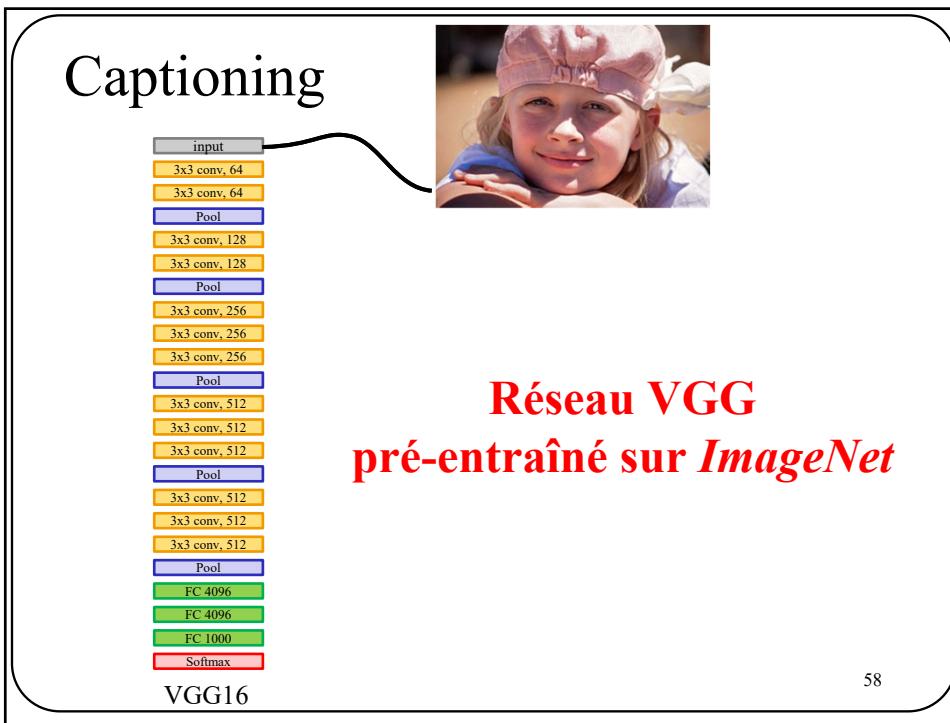
M entrées et N sorties



56

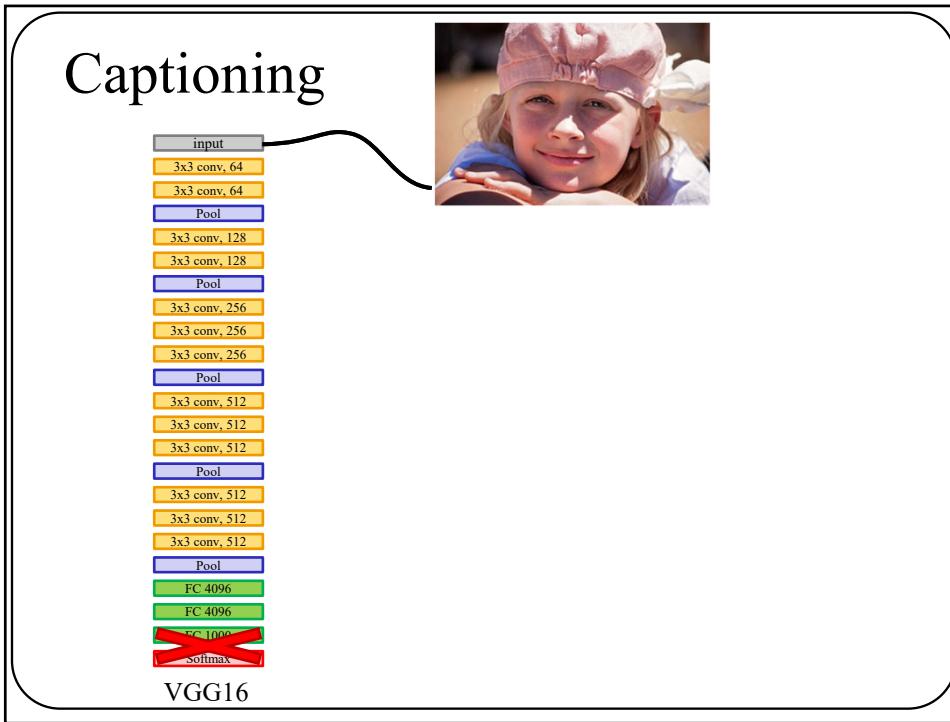


57

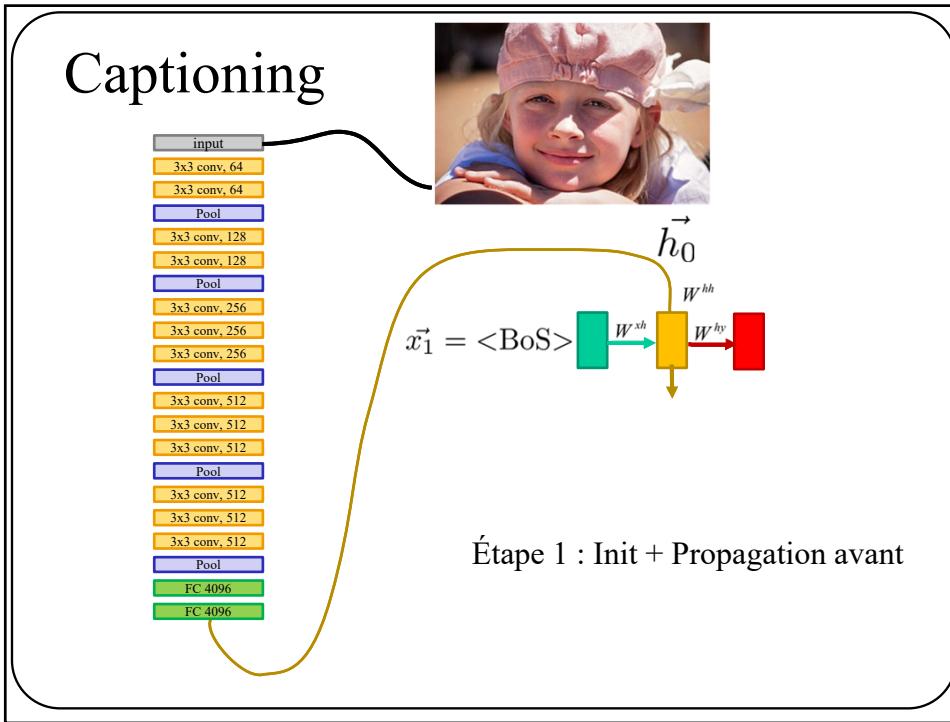


58

58

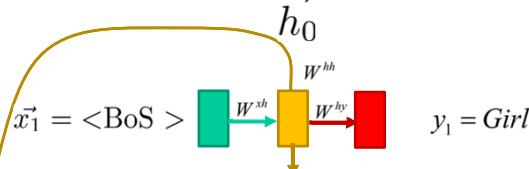
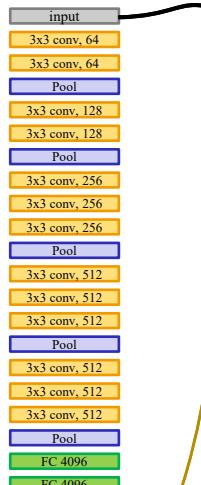


59



60

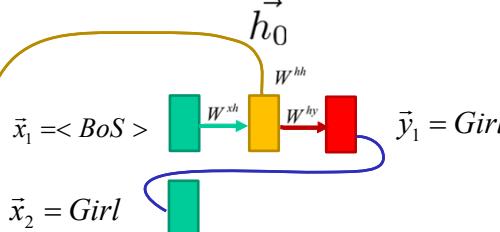
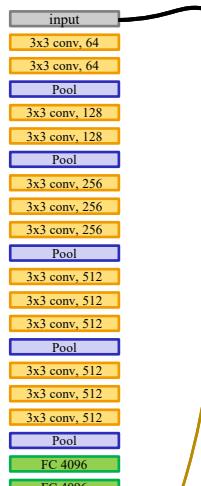
Captioning



Étape 2 : Échantillonage

61

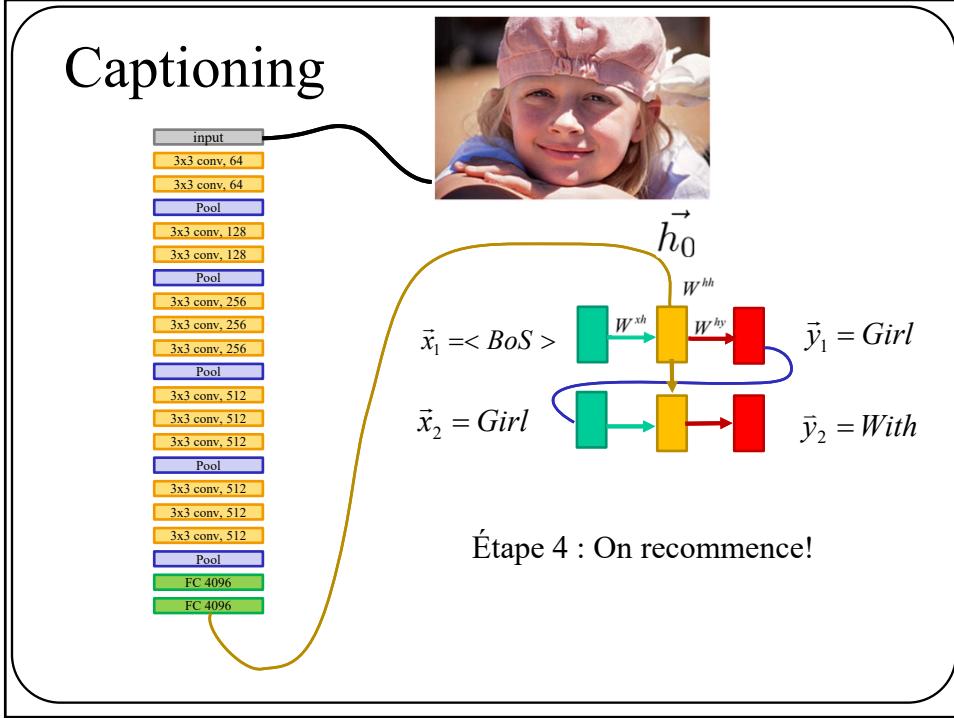
Captioning



Étape 3 : Copie la sortie vers l'entrée

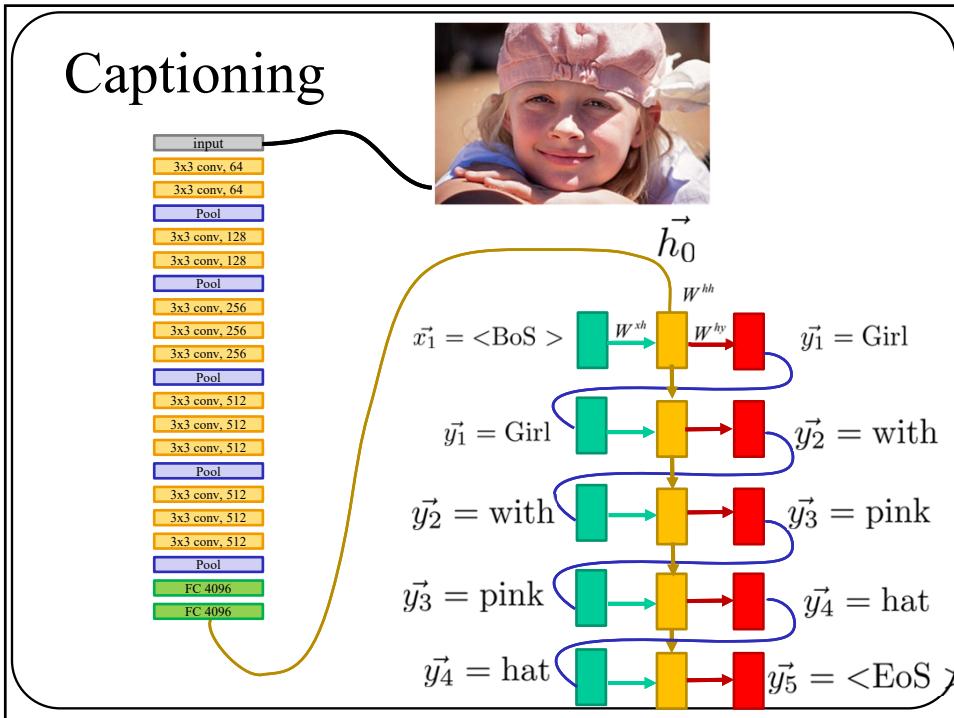
62

Captioning



63

Captioning



64

Exemples de résultats

https://www.vision-objects.com/api/v2/jt/xce1frp_2ndusdwk/2ghxudodm5



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



a man riding a wave on top of a surfboard



a group of people playing a game with nintendo wii controllers



a couple of zebra standing on top of a dirt field



a street sign on a pole in front of a building

65

65

Exemples d'erreurs

https://www.vision-objects.com/api/v2/jt/xce1frp_2ndusdwk/2ghxudodm5



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

66

66

NeuralTalk and Walk [kws v=2yp hr1frp 24797<5334](#)



67

67

Analyse de texte

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

Pour des **caractères**...

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

...

68

68

Analyse de texte

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

Pour des **mots**...

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

...

69

69

Prédiction sur des lettres vs. mots

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

...

...

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

...

70

Prédiction sur des lettres vs. mots

'a'=[1,0,0,...,0]

'b'=[0,1,0,...,0]

'c'=[0,0,1,...,0]

...

'grand'=[0,0,...,0,1,...,0]

'granden'=[0,0,...,0,0,...,0]

'grandeu'=[0,0,...,0,0,...,0]

...

En analyse des langues, un vecteur numérique associé à une séquence de caractères se nomme « JETON » (« token »)

lettres

mots

71

Limites des Jetons « one-hot »

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

1- Peu efficace en mémoire lorsque non compressés

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

2- Pas de distance sémantique entre les Jetons:

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')]

74

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

Exemple jouet: on veut représenter ces 8 mots par des jetons à 4 éléments

	Jeton « one-hot »								Dictionnaire de Jetons			
‘the’	1	0	0	0	0	0	0	0	2	3	4	5
‘quick’	0	1	0	0	0	0	0	0	-1	-3	-2	2
‘brown’	0	0	1	0	0	0	0	0	11	6	4	-3
‘fox’	0	0	0	1	0	0	0	0	-4	8	-4	4
‘jumps’	0	0	0	0	1	0	0	0	24	-6	42	17
‘over’	0	0	0	0	0	1	0	0	91	13	14	-5
‘lazy’	0	0	0	0	0	0	1	0	0	36	4	56
‘dog’	0	0	0	0	0	0	0	1	-1	0	1	35



75

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage

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

Ex: sélectionner le jeton de « brown »

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

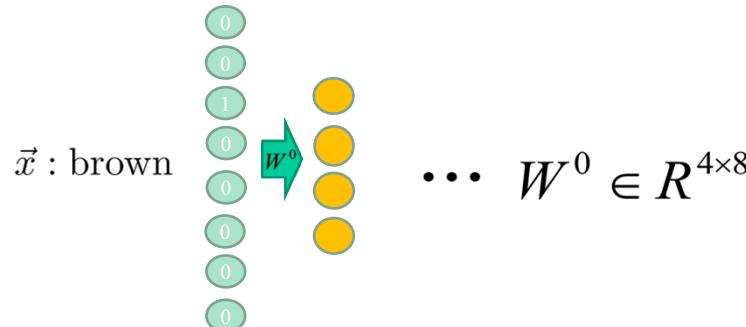
76

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire jeton = matrice d'encodage

Première couche d'un réseau de neurones

=
matrice d'encodage



77

Word2Vec s'appuie sur 2 idées fondamentales

Idée 1: Dictionnaire = matrice d'encodage

Première couche d'un réseau de neurones

=
matrice d'encodage



$$\text{jeton}_{\vec{x}} = W^0 \vec{x}$$

78

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

79

Word2Vec s'appuie sur 2 idées fondamentales

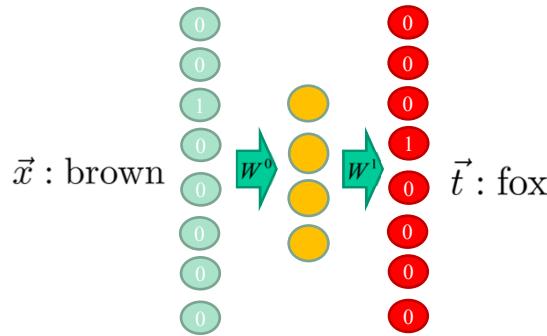
Idée 2: 2 mots proches dans un texte = 2 mots proches sémantiquement

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**

80

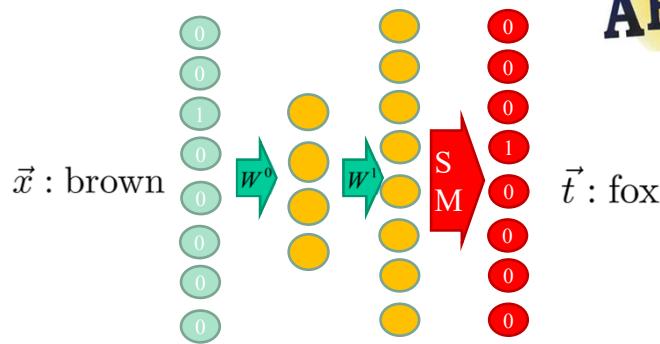
Word2Vec [Mikolov et al. '13]



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

81

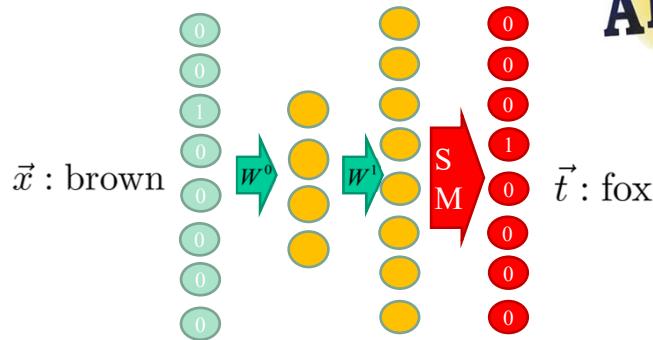
Word2Vec [Mikolov et al. '13]



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

82

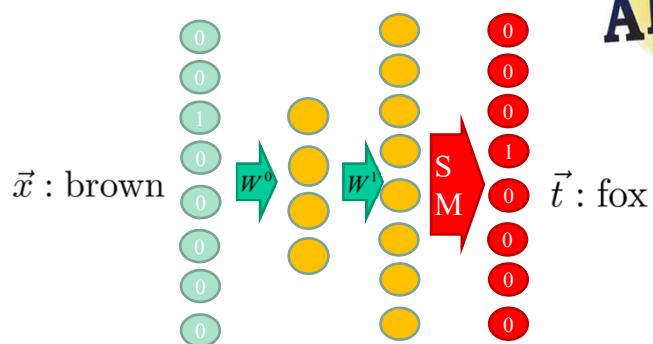
Word2Vec [Mikolov et al. '13]



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

83

Word2Vec [Mikolov et al. '13]



Lorsqu'entraîné, utiliser W^0
comme dictionnaire

84

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

85

Distance sémantique entre deux mots = distance entre leur jeton

Word	First similar word	Second similar word	Third similar word
colosseum	rome (0.994)	roma (0.994)	coliseum (0.994)
colosseo	anfiteatro (0.995)	travel (0.994)	italia (0.994)
scala	aux (0.993)	camelias (0.992)	milano (0.992)
pompei	retweeted (0.988)	nuovi (0.979)	settembre (0.978)
roma	rome (0.995)	metro (0.994)	colosseum (0.994)
italia	anfiteatro (0.995)	rome (0.995)	colosseo (0.994)
italy	travel (0.998)	davanti (0.997)	photography (0.997)

Word	Similar Words	Similarity	Word	Similar Words	Similarity
Linux	windows	0.85	Twitter	facebook	0.90
	redhat	0.83		instagram	0.86
	unix	0.83		netflix	0.84
	mac os	0.82		snapchat	0.82
	citrix	0.81		google	0.81
	serveurs	0.80		tweets	0.80
	microsoft	0.79		youtube	0.80
	ibm	0.79		linkedin	0.77
windows server	windows server	0.79		maddyness	0.77
	env windows	0.79		tweet	0.77

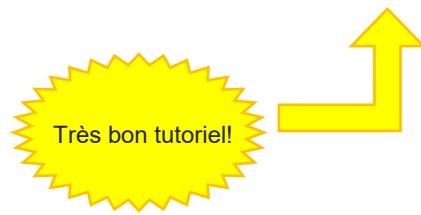
Ahmia, Oussama & Béchet, Nicolas & Marteau, Pierre-Francois. *Two Multilingual Corpora Extracted from the Tenders Electronic Daily for Machine Learning and Machine Translation Applications*.in LREC 2018

86

86

Word2Vec

<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

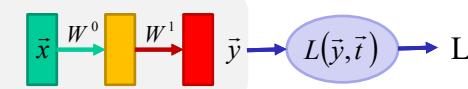
87

Comment entraîner un RNN?

88

Histoire de gradients

RN de classification avec entropie croisée



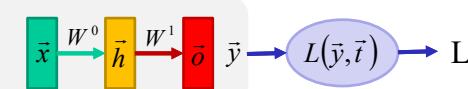
$$\vec{y}(\vec{x}) = S_M(W^1 \tanh(W^0 \vec{x}))$$

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

89

Histoire de gradients

Simple RN de classification avec entropie croisée



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

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

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

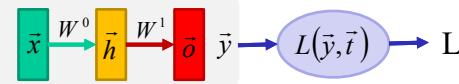
$$L = L_{CE}(\vec{y}, \vec{t})$$

Propagation
avant

90

Histoire de gradients

Simple RN de classification avec entropie croisée



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

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

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

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

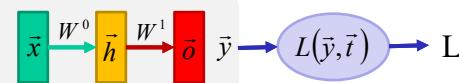
Pour entraîner le réseau
il faut calculer

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

91

Histoire de gradients

Simple RN de classification avec entropie croisée



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

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

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

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

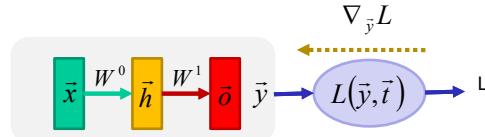
Dérivée en chaîne

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

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

92

Histoire de gradients



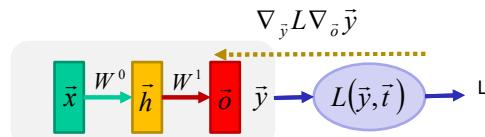
$$\begin{aligned}\vec{h} &= \tanh(W^0 \vec{x}) \\ \vec{o} &= W^1 \vec{h} \\ \vec{y} &= S_M(\vec{o}) \\ L &= L_{CE}(\vec{y}, \vec{t})\end{aligned}$$

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

↓ ↑
Rétro-propagation

93

Histoire de gradients



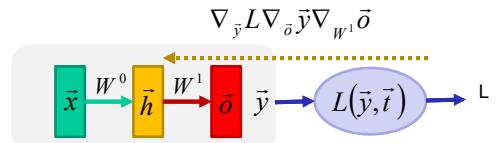
$$\begin{aligned}\vec{h} &= \tanh(W^0 \vec{x}) \\ \vec{o} &= W^1 \vec{h} \\ \vec{y} &= S_M(\vec{o}) \\ L &= L_{CE}(\vec{y}, \vec{t})\end{aligned}$$

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

↓ ↑
Rétro-propagation

94

Histoire de gradients



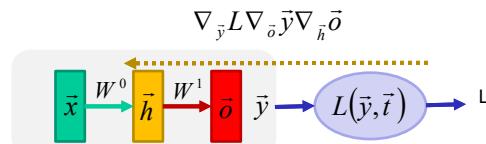
$$\begin{aligned}
 \vec{h} &= \tanh(W^0 \vec{x}) \\
 \vec{o} &= W^1 \vec{h} \\
 \vec{y} &= S_M(\vec{o}) \\
 L &= L_{CE}(\vec{y}, \vec{t})
 \end{aligned}
 \quad \downarrow \quad \uparrow$$

$$\begin{aligned}
 \nabla_{W^1} \vec{o} &= \vec{h} \\
 \nabla_{\vec{o}} \vec{y} &= I \vec{y}^T - \vec{y}^T \vec{y} \\
 \nabla_{\vec{y}} L &= -\frac{\vec{t}}{\vec{y}}
 \end{aligned}$$

Rétro-propagation

95

Histoire de gradients



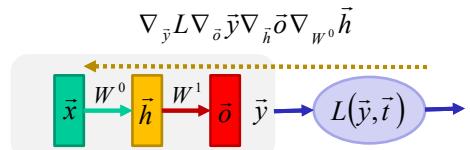
$$\begin{aligned}
 \vec{h} &= \tanh(W^0 \vec{x}) \\
 \vec{o} &= W^1 \vec{h} \\
 \vec{y} &= S_M(\vec{o}) \\
 L &= L_{CE}(\vec{y}, \vec{t})
 \end{aligned}
 \quad \downarrow \quad \uparrow$$

$$\begin{aligned}
 \nabla_{\vec{h}} \vec{o} &= W^1 \\
 \nabla_{W^1} \vec{o} &= \vec{h} \\
 \nabla_{\vec{o}} \vec{y} &= I \vec{y}^T - \vec{y}^T \vec{y} \\
 \nabla_{\vec{y}} L &= -\frac{\vec{t}}{\vec{y}}
 \end{aligned}$$

Rétro-propagation

96

Histoire de gradients

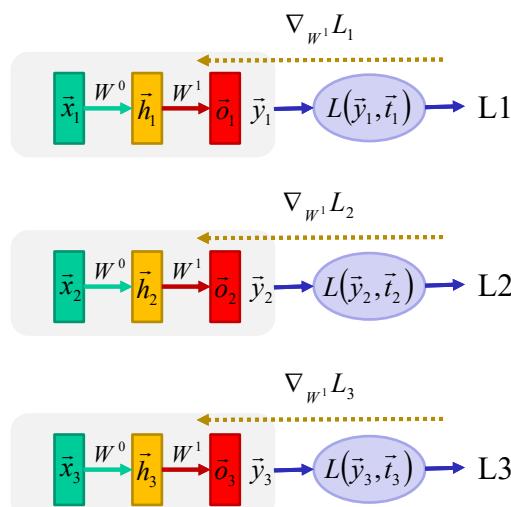


$$\begin{aligned}
 \vec{h} &= \tanh(W^0 \vec{x}) \\
 \vec{o} &= W^1 \vec{h} \\
 \vec{y} &= S_M(\vec{o}) \\
 L &= L_{CE}(\vec{y}, \vec{t})
 \end{aligned}
 \quad \downarrow \quad \begin{aligned}
 \nabla_{W^0} \vec{h} &= 1 - \tanh^2(W^1 \vec{x}) \vec{x} \\
 \nabla_{\vec{h}} \vec{o} &= W^1 \\
 \nabla_{W^1} \vec{o} &= \vec{h} \\
 \nabla_{\vec{o}} \vec{y} &= I \vec{y}^T - \vec{y}^T \vec{y} \\
 \nabla_{\vec{y}} L &= -\frac{\vec{t}}{\vec{y}}
 \end{aligned}$$

Rétro-propagation

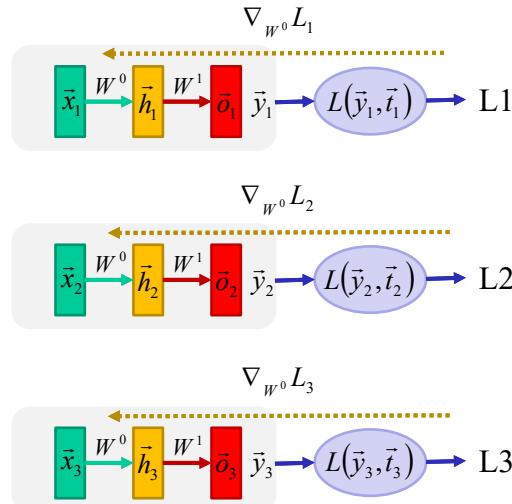
97

Ex.: 3 données, 3 rétro-propagations



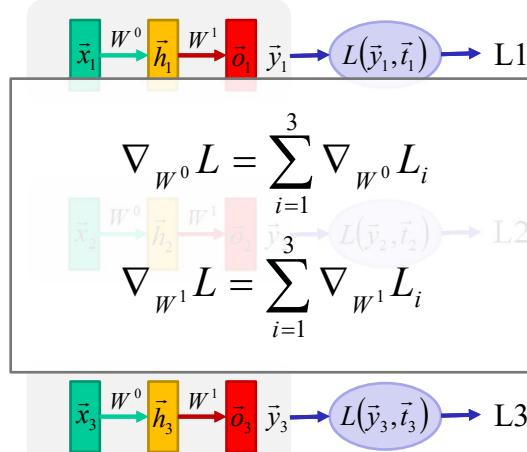
98

Ex.: 3 données, 3 rétro-propagations



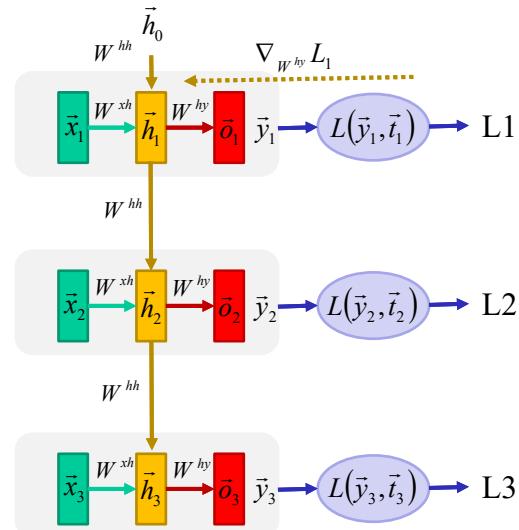
99

3 rétro-propagations



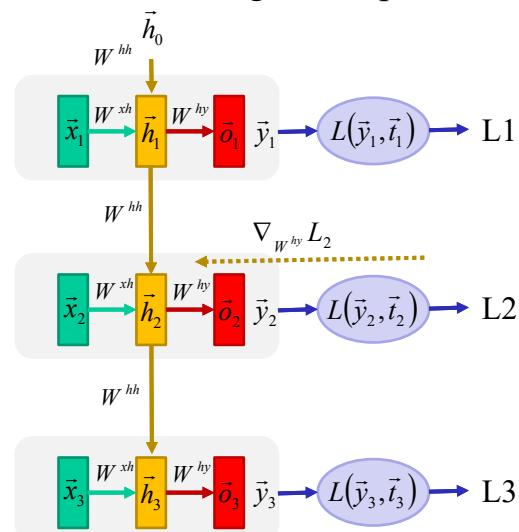
100

Réseau récurrent: gradient pour W^{hy}



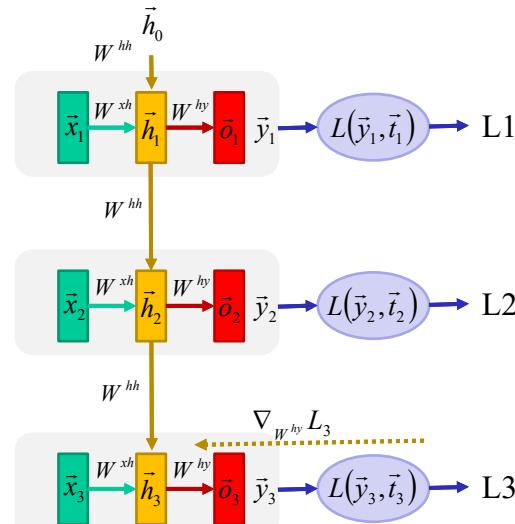
101

Réseau récurrent: gradient pour W^{hy}



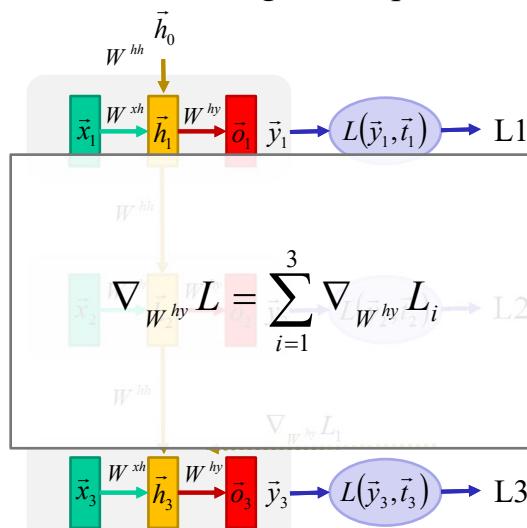
102

Réseau récurrent: gradient pour W^{hy}



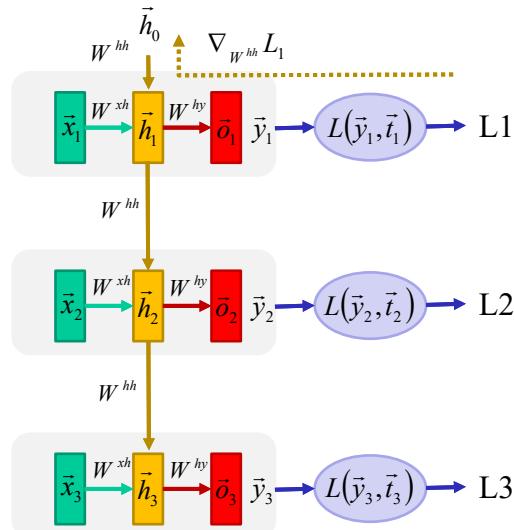
103

Réseau récurrent: gradient pour W^{hy}



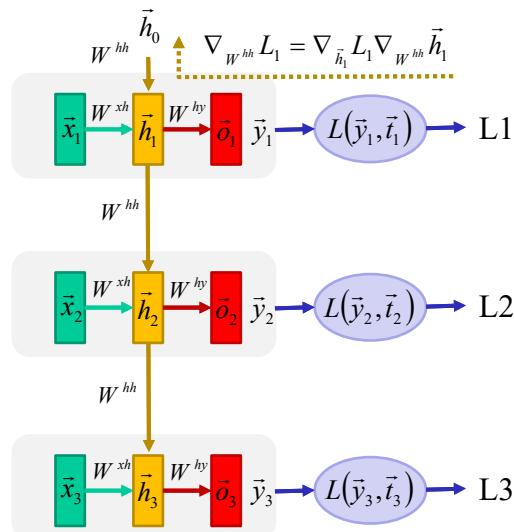
104

Réseau récurrent: gradient pour W^{hh}



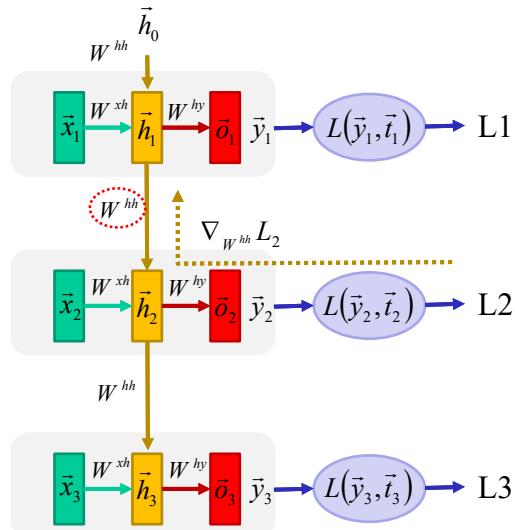
105

Réseau récurrent: gradient pour W^{hh}



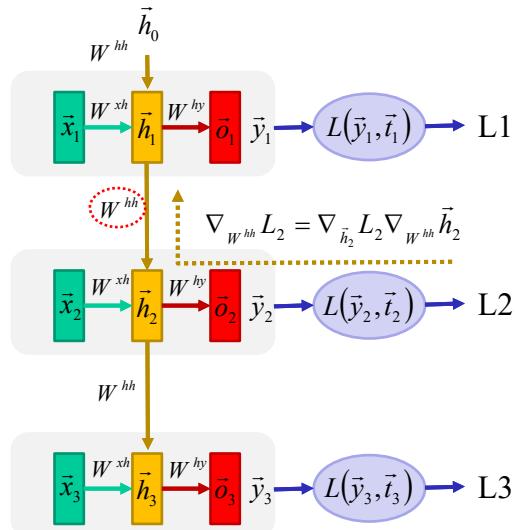
106

Réseau récurrent: gradient pour W^{hh}



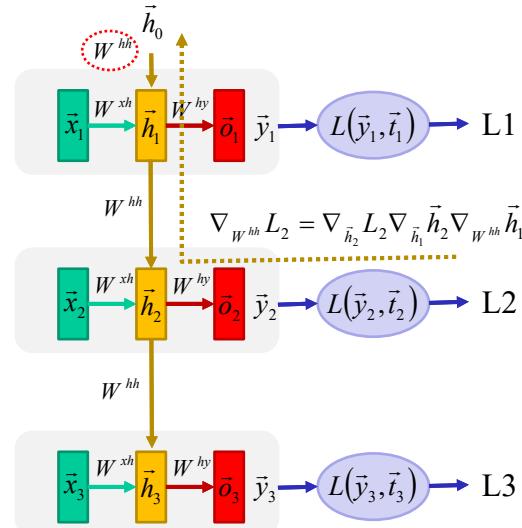
107

Réseau récurrent: gradient pour W^{hh}



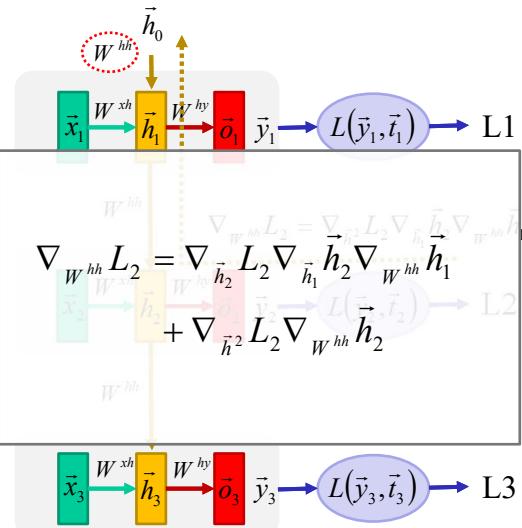
108

Réseau récurrent: gradient pour W^{hh}



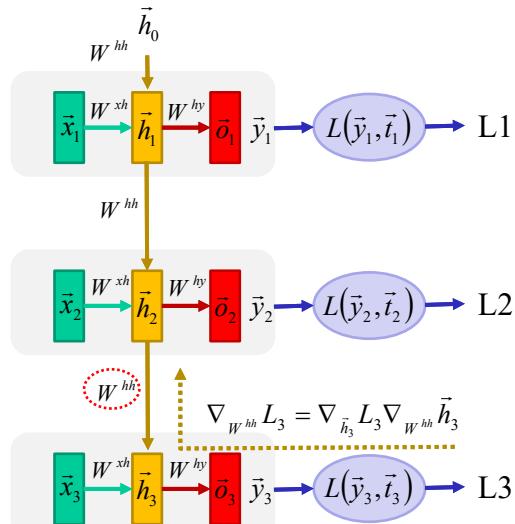
109

Réseau récurrent: gradient pour W^{hh}



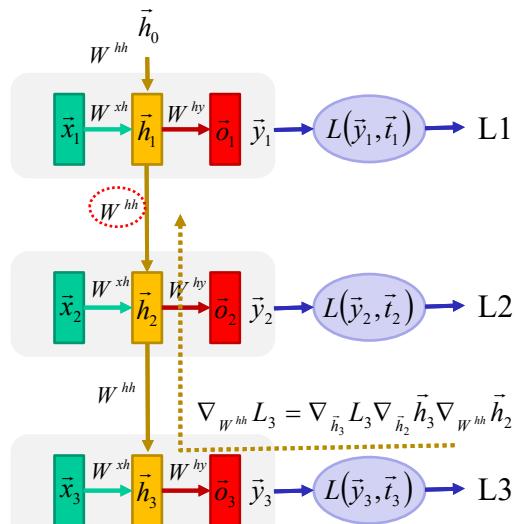
110

Réseau récurrent: gradient pour W^{hh}



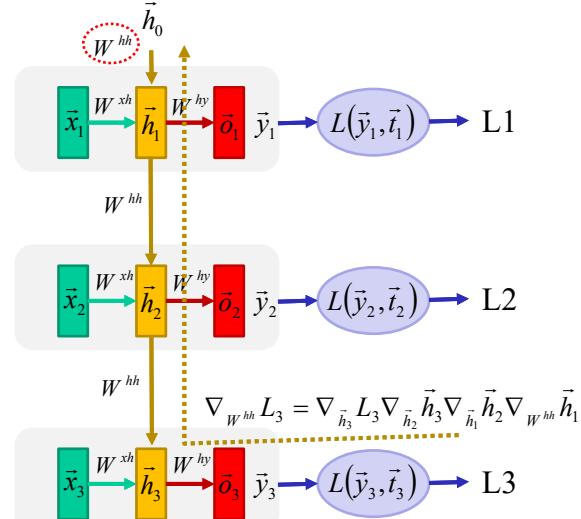
111

Réseau récurrent: gradient pour W^{hh}



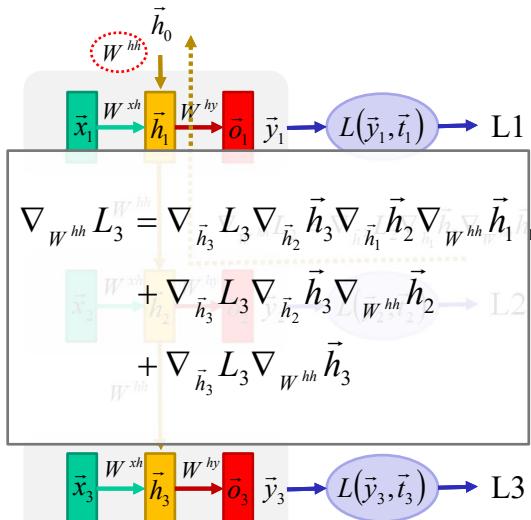
112

Réseau récurrent: gradient pour W^{hh}



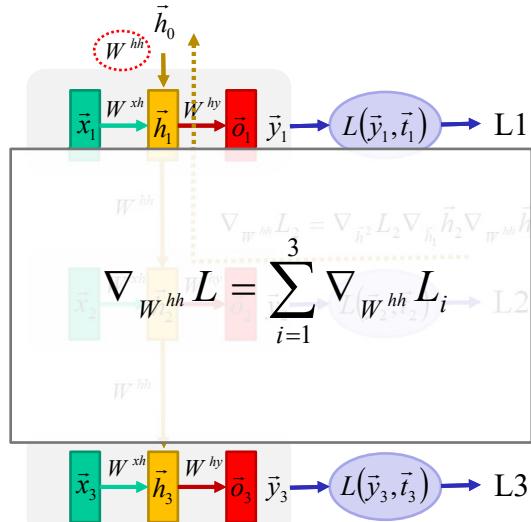
113

Réseau récurrent: gradient pour W^{hh}



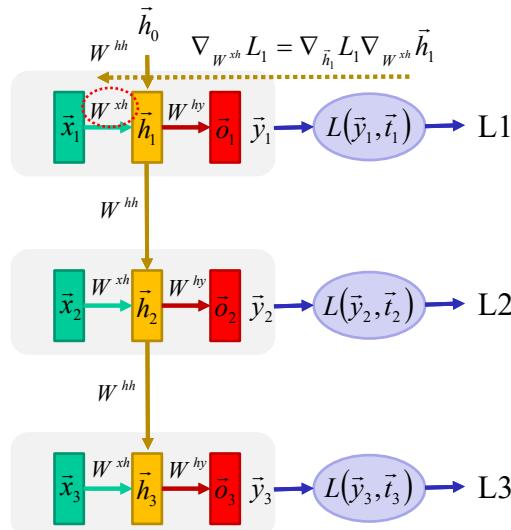
114

Réseau récurrent: gradient pour W^{hh}



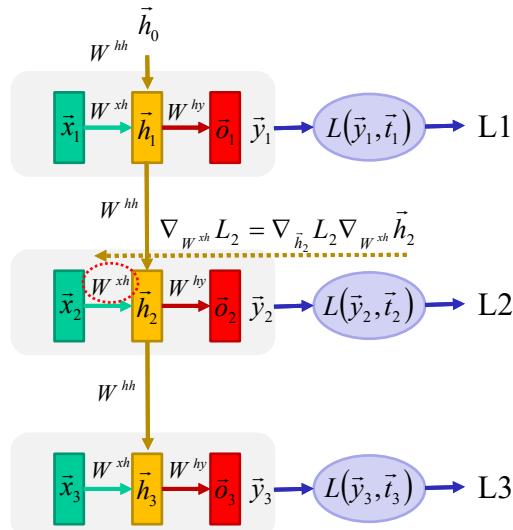
116

Réseau récurrent: gradient pour W^{xh}



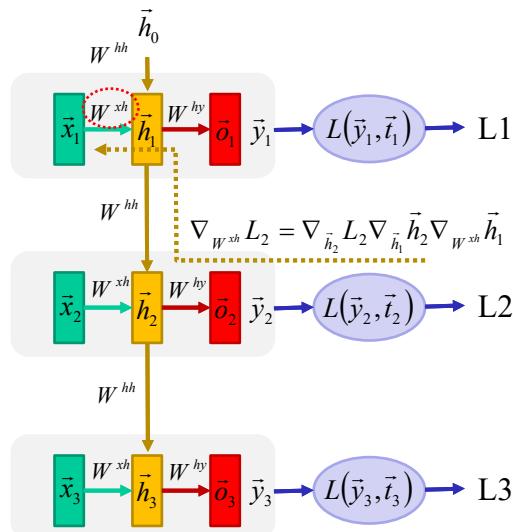
118

Réseau récurrent: gradient pour W^{xh}



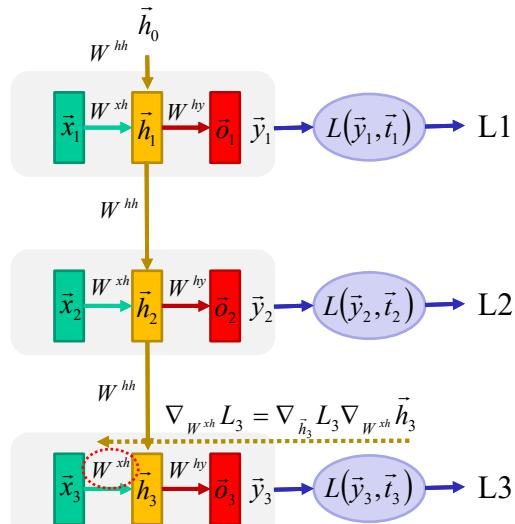
119

Réseau récurrent: gradient pour W^{xh}



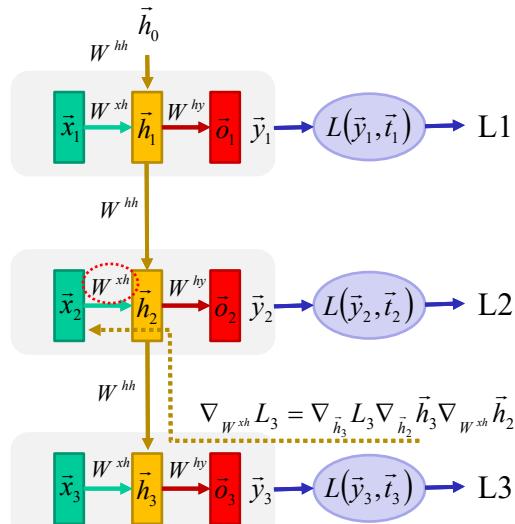
120

Réseau récurrent: gradient pour W^{xh}



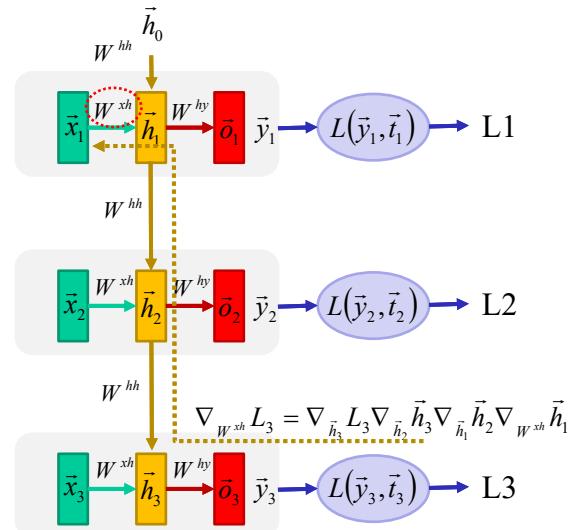
121

Réseau récurrent: gradient pour W^{xh}



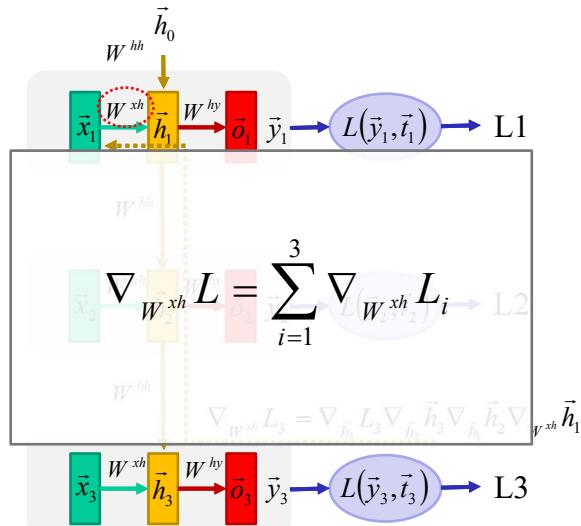
122

Réseau récurrent: gradient pour W^{xh}



123

Réseau récurrent: gradient pour W^{xh}



124

Réseau récurrent: calcul du gradient

Moins difficile qu'il n'y paraît.

```

43     # backward pass: compute gradients going backwards
44     dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
45     dbh, dyb = np.zeros_like(bh), np.zeros_like(by)
46     dhnext = np.zeros_like(hs[0])
47
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         dyb += 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
60     for dparam in [dWxh, dWhh, dWhy, dbh, dyb]:
61         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
62
63     return loss, dWxh, dWhh, dWhy, dbh, dyb, hs[len(inputs)-1]

```

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

125

Les réseaux récurrents ont un
inconvénient majeur:

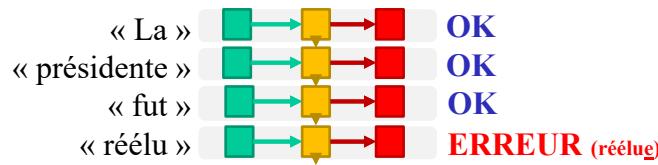
difficile à établir des
relations à longue distance

126

126

Exemples: analyse grammaticale

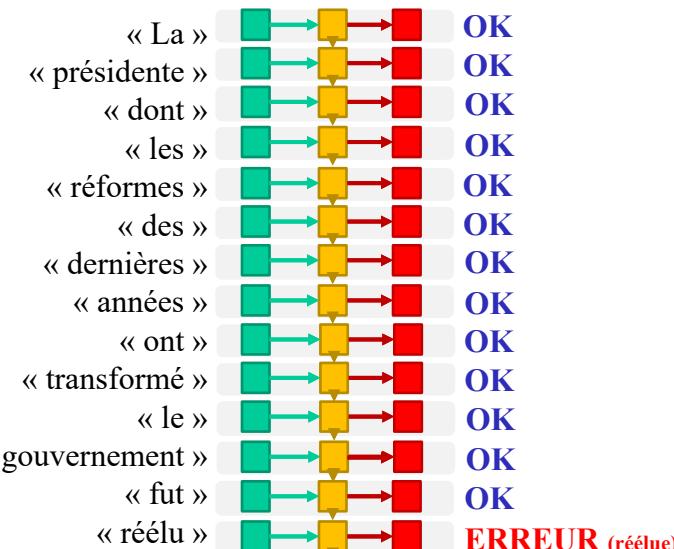
Entraîner un réseau à détecter des erreurs grammaticales



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

127

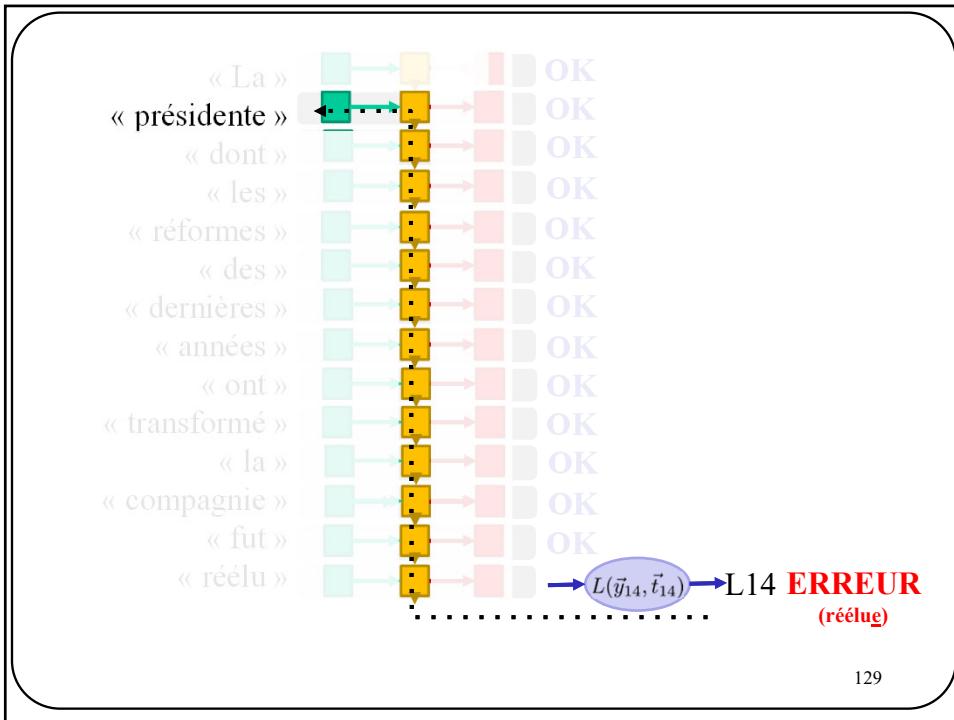
127



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

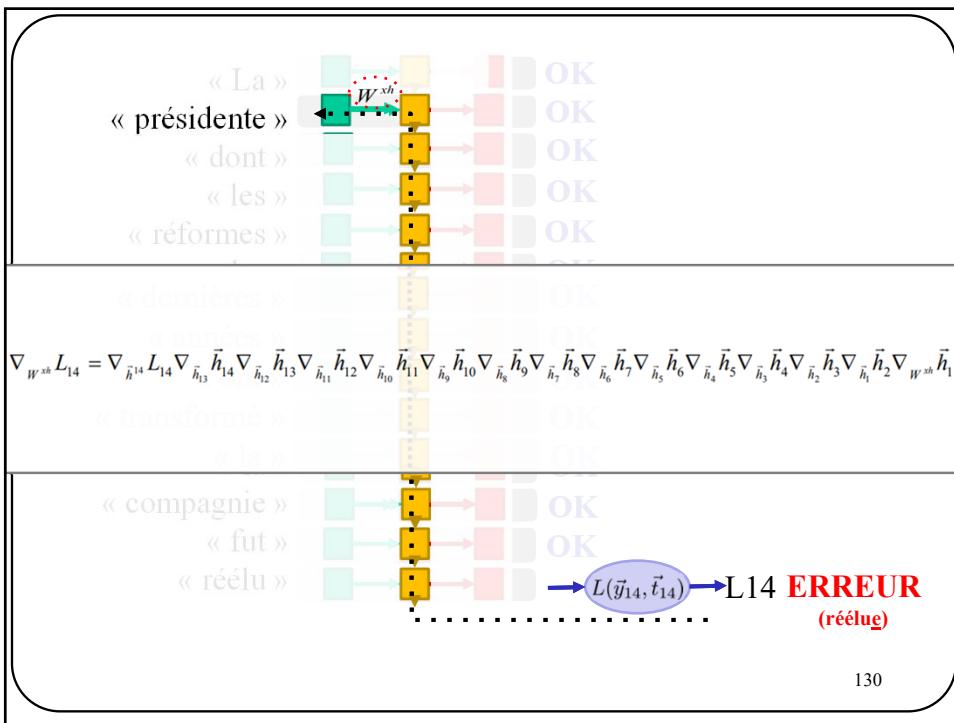
128

128



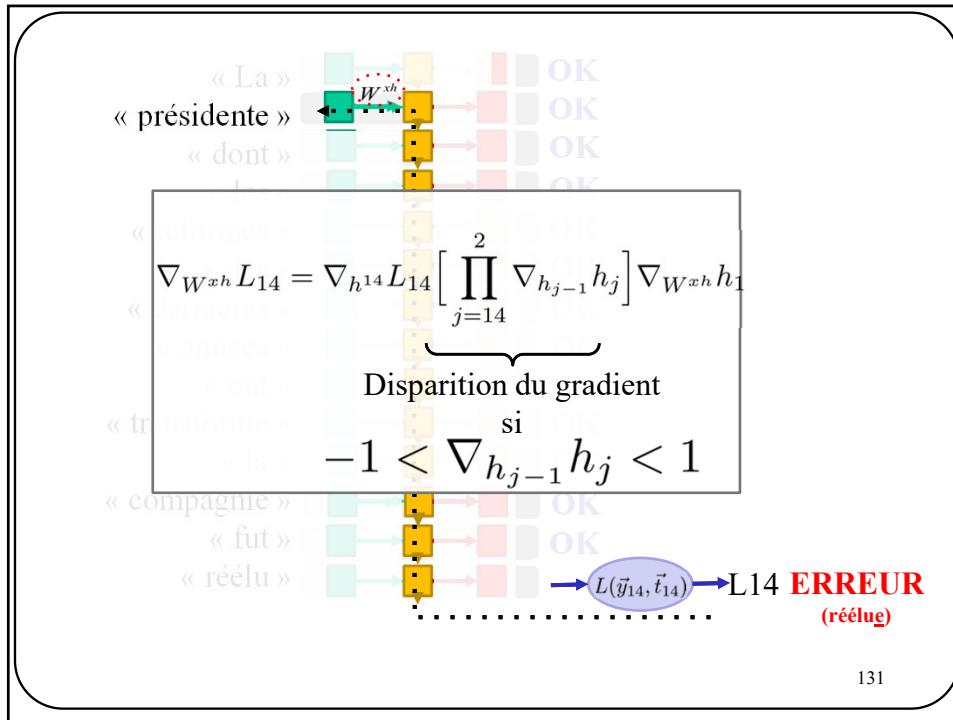
129

129

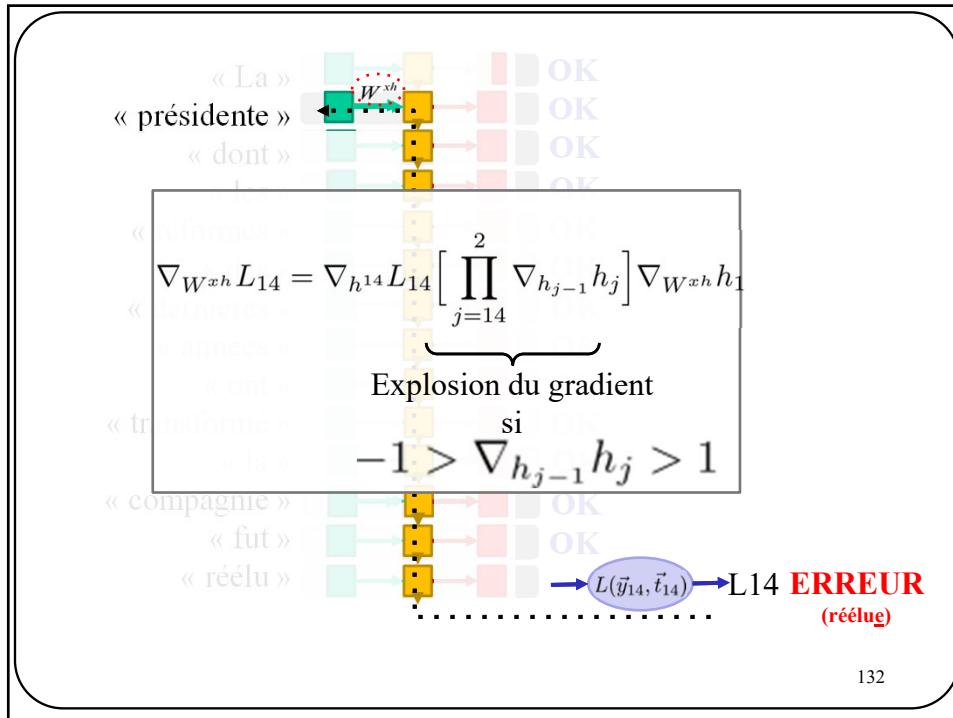


130

130



131



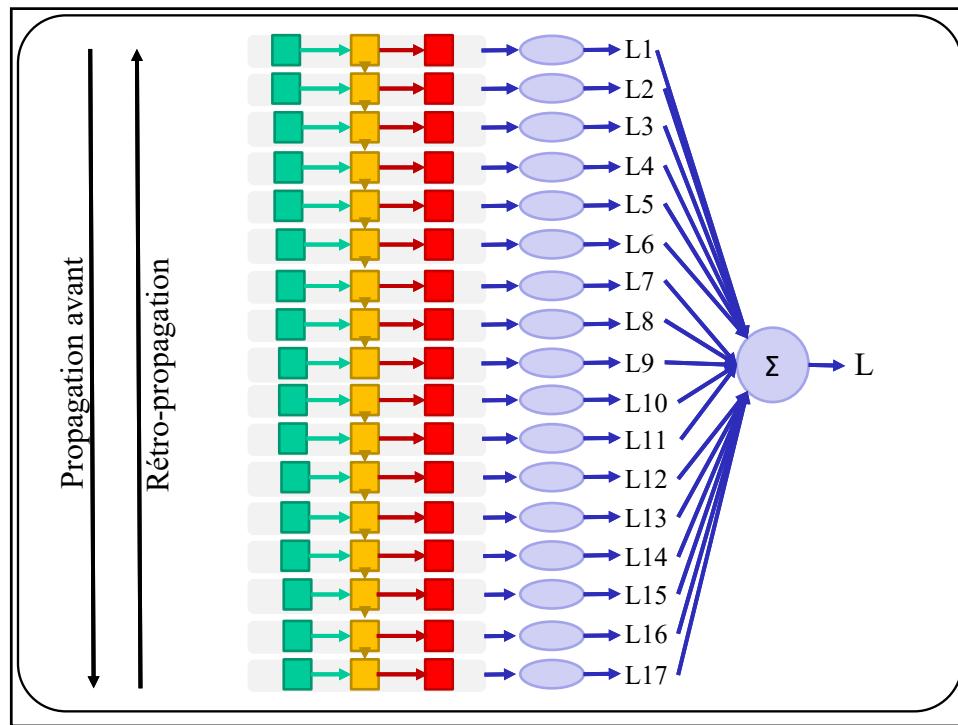
132

Problème connexe

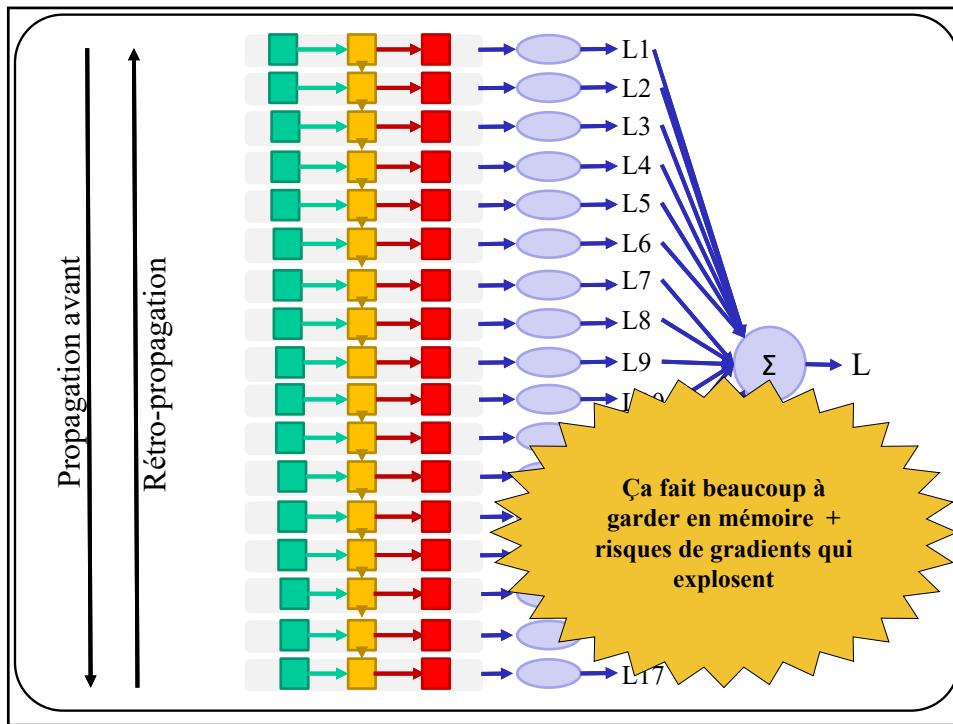
Gestion de la mémoire

133

133



134

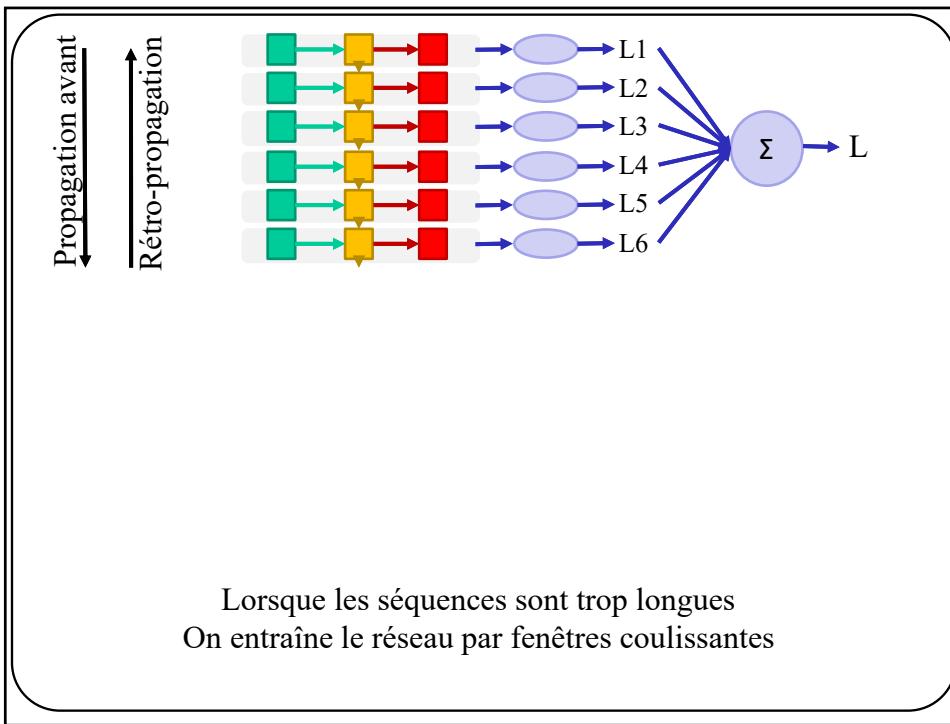


135

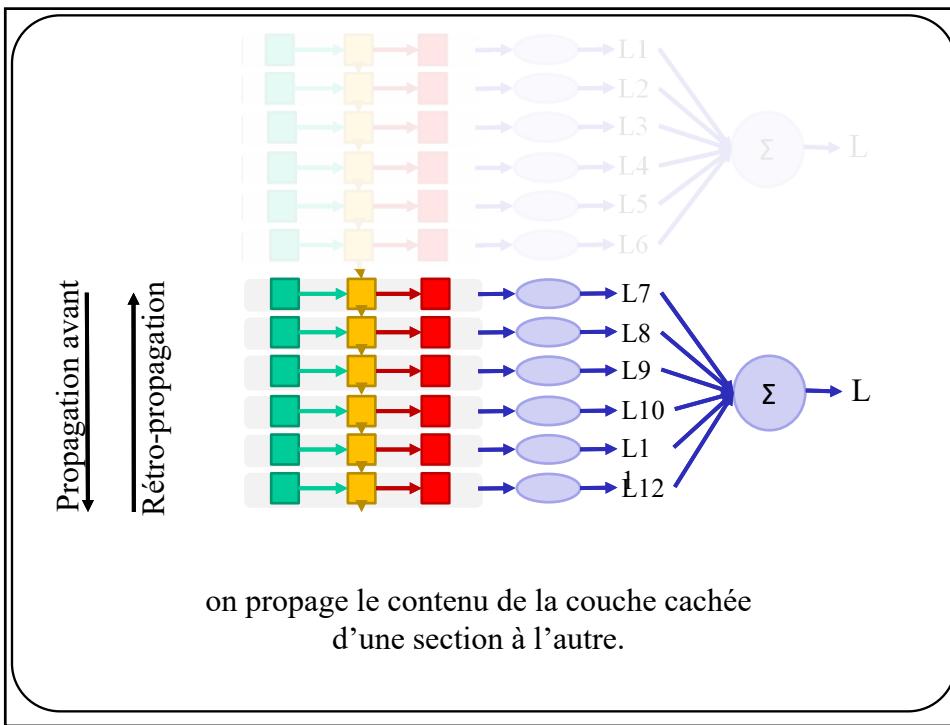
Solution pour la gestion de la
mémoire
Fenêtres coulissantes

136

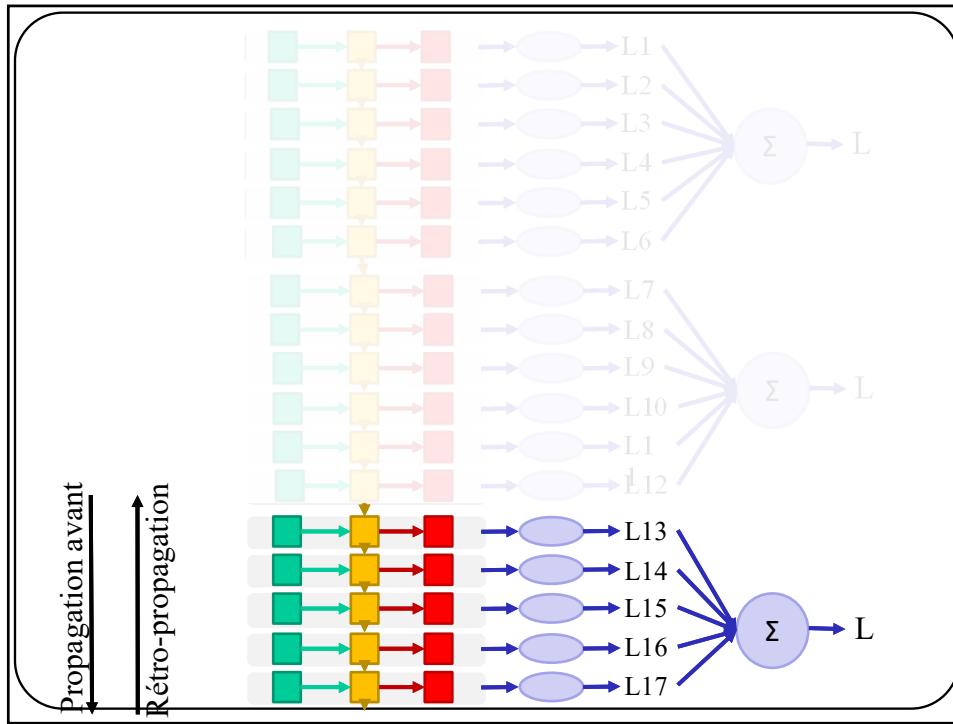
136



137



138



139

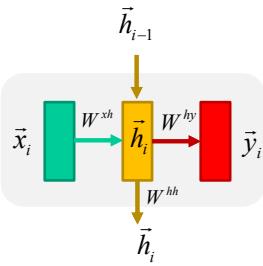
Solution à la disparition du gradient:

Gated Recurrent Unit : GRU
Long-Short Term Memory : LSTM

140

140

Illustration + formulation d'un RNN

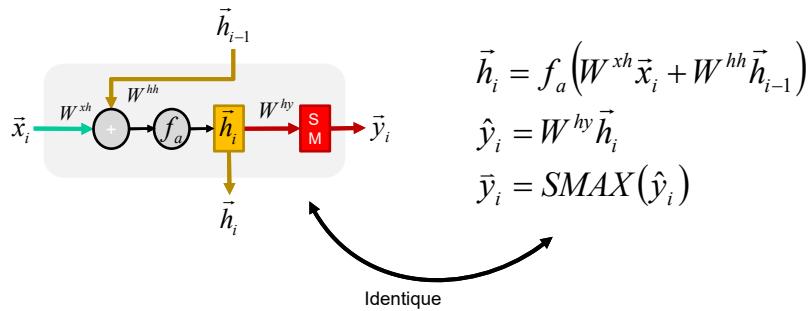


$$\begin{aligned}\vec{h}_i &= f_a(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \hat{y}_i &= W^{hy}\vec{h}_i \\ \vec{y}_i &= SMAX(\hat{y}_i)\end{aligned}$$

141

141

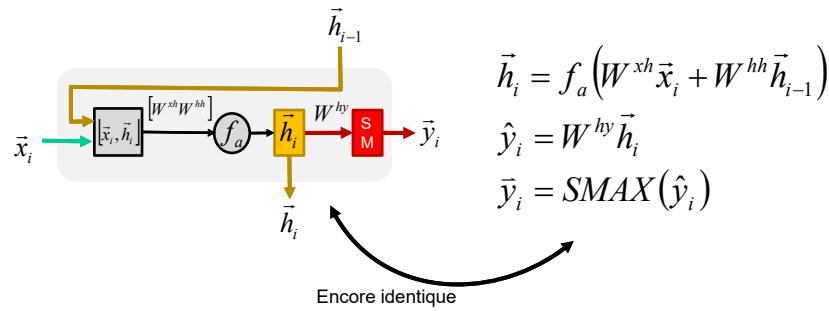
Autre illustration du même RNN



142

142

Autre illustration du même RNN



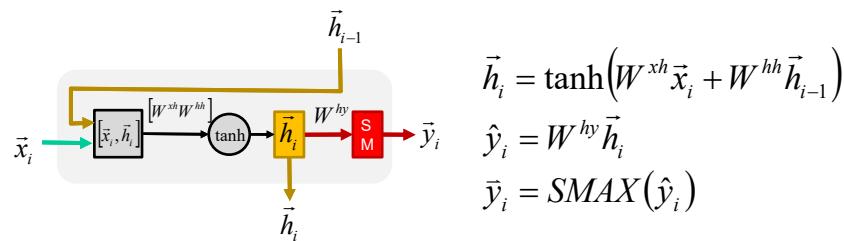
143

143

GRU (Gated Recurrent Unit)

Modif 1

$$f_a = \tanh$$



144

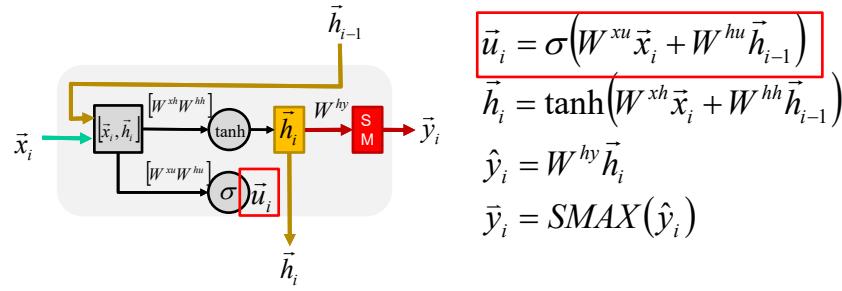
144

GRU (Gated Recurrent Unit)

Modif 2

Update gate

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



$$\begin{aligned}\vec{u}_i &= \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1}) \\ \vec{h}_i &= \tanh(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1}) \\ \hat{y}_i &= W^{hy} \vec{h}_i \\ \bar{y}_i &= \text{SMAX}(\hat{y}_i)\end{aligned}$$

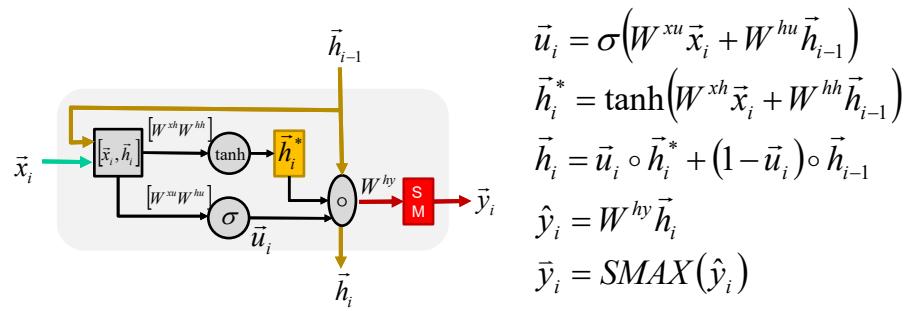
145

145

GRU (Gated Recurrent Unit)

Modif 2

Update gate



$$\begin{aligned}\vec{u}_i &= \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1}) \\ \vec{h}_i^* &= \tanh(W^{xh} \vec{x}_i + W^{hh} \vec{h}_{i-1}) \\ \vec{h}_i &= \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1} \\ \hat{y}_i &= W^{hy} \vec{h}_i \\ \bar{y}_i &= \text{SMAX}(\hat{y}_i)\end{aligned}$$

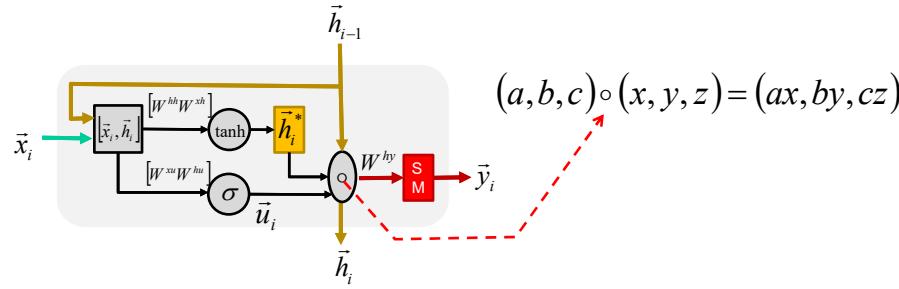
146

146

GRU (Gated Recurrent Unit)

Modif 2

« element-wise product »
ou
« Hadamard product »



147

147

GRU (Gated Recurrent Unit)

Modif 3

Reset gate

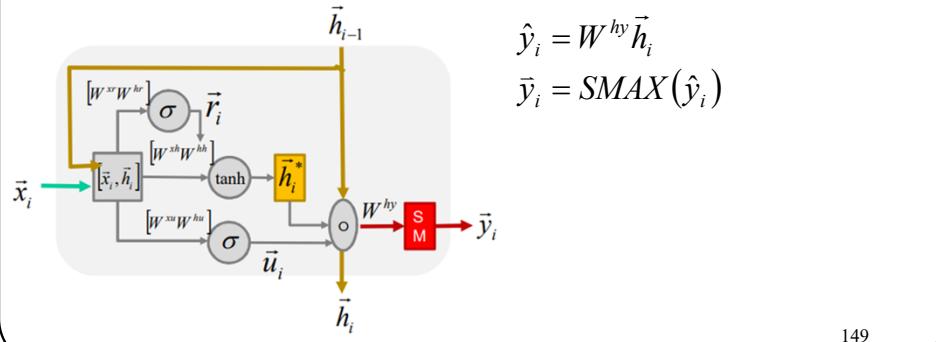
$$\begin{aligned}\vec{u}_i &= \sigma(W^{xu} \vec{x}_i + W^{hu} \vec{h}_{i-1}) \\ \vec{r}_i &= \sigma(W^{xr} \vec{x}_i + W^{hr} \vec{h}_{i-1}) \\ \vec{h}_i^* &= \tanh(W^{xh} \vec{x}_i + W^{hh} (\vec{r}_i \circ \vec{h}_{i-1})) \\ \vec{h}_i &= \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1} \\ \hat{y}_i &= W^{hy} \vec{h}_i \\ \bar{y}_i &= SMAX(\hat{y}_i)\end{aligned}$$

148

148

Comprendre les *gates*

$$SI \quad \begin{matrix} u_i \\ r_i \end{matrix} = 1 \quad \left\{ \begin{array}{l} \vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\ \vec{r}_i = \sigma(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}) \\ \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}(\vec{r}_i \circ \vec{h}_{i-1})) \\ \vec{h}_i = \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1} \end{array} \right.$$

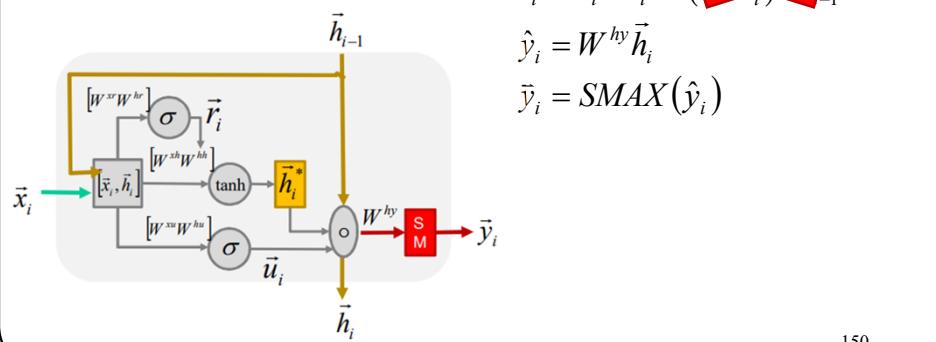


149

149

Comprendre les *gates*

$$SI \quad \begin{matrix} u_i \\ r_i \end{matrix} = 1 \quad \left\{ \begin{array}{l} \vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\ \vec{r}_i = \sigma(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}) \\ \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}(\vec{r}_i \circ \vec{h}_{i-1})) \\ \vec{h}_i = \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1} \end{array} \right.$$

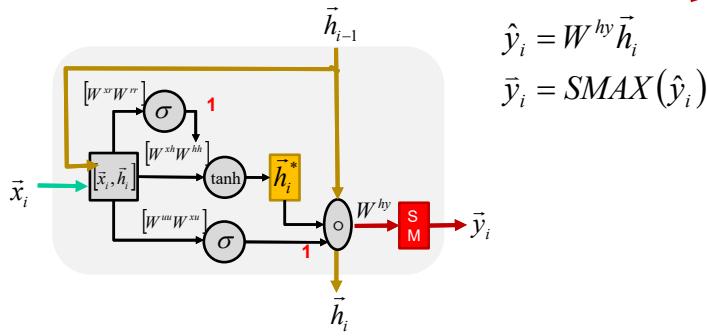


150

150

Comprendre les *gates*

$$SI \quad \begin{cases} u_i = 1 \\ r_i = 1 \end{cases} \quad \left\{ \begin{array}{l} \vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\ \vec{r}_i = \sigma(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}) \\ \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \vec{h}_i = \vec{r}_i \circ \vec{h}_i^* + (1 - \vec{r}_i) \circ \vec{h}_{i-1} \end{array} \right.$$

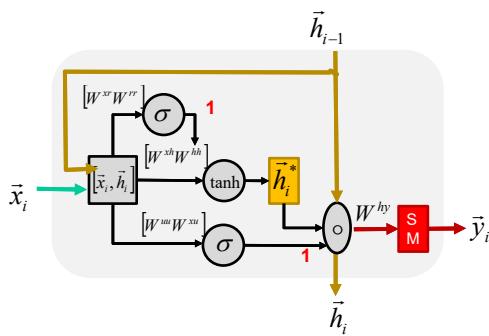


151

151

Comprendre les *gates*

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

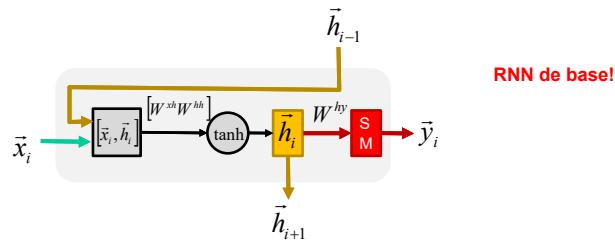


152

152

Comprendre les *gates*

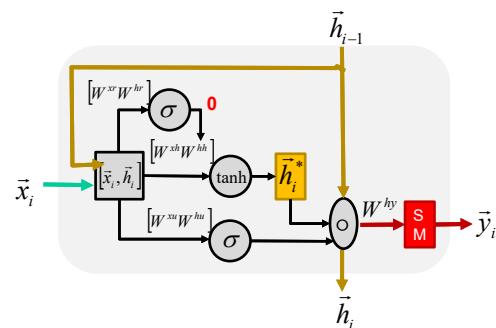
$$SI \quad \begin{cases} \vec{u}_i = 1 \\ \vec{r}_i = 1 \end{cases} \quad \begin{cases} \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \vec{h}_i = \vec{h}_i^* \\ \hat{y}_i = W^{hy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i) \end{cases}$$



153

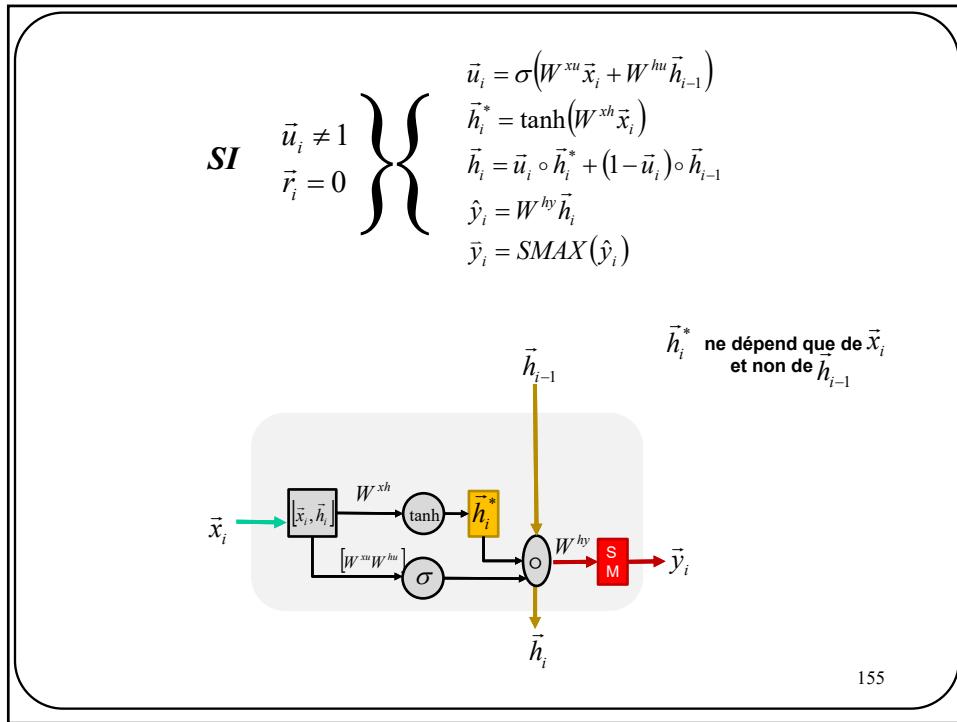
153

$$SI \quad \begin{cases} \vec{u}_i \neq 1 \\ \vec{r}_i = 0 \end{cases} \quad \begin{cases} \vec{u}_i = \sigma(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}) \\ \vec{r}_i = \sigma(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}) \\ \vec{h}_i^* = \tanh(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}) \\ \vec{h}_i = \vec{u}_i \circ \vec{h}_i^* + (1 - \vec{u}_i) \circ \vec{h}_{i-1} \\ \hat{y}_i = W^{hy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i) \end{cases}$$

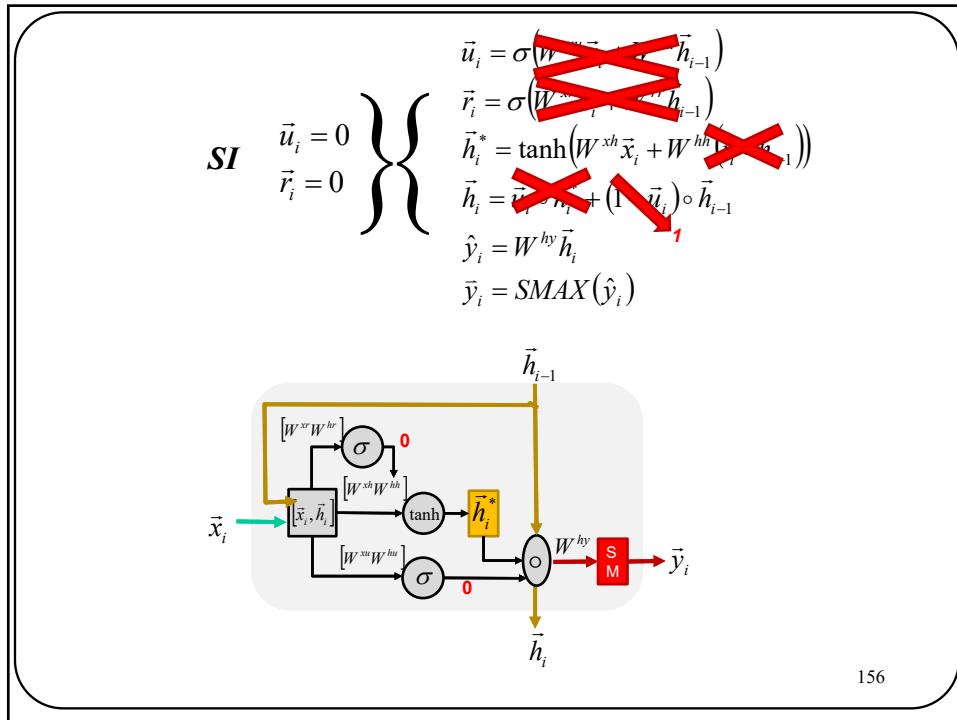


154

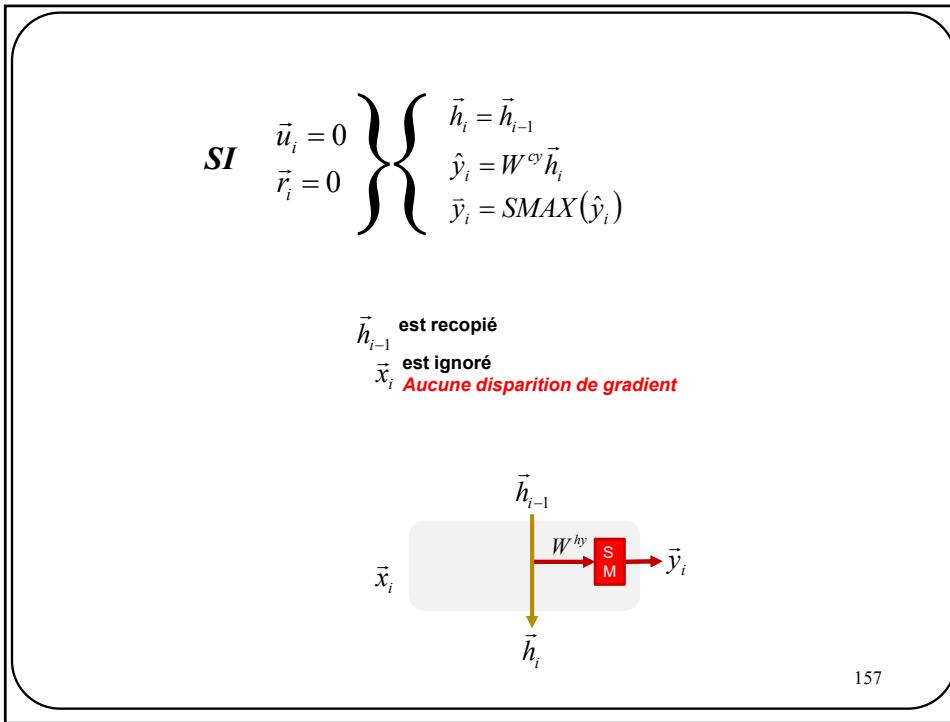
154



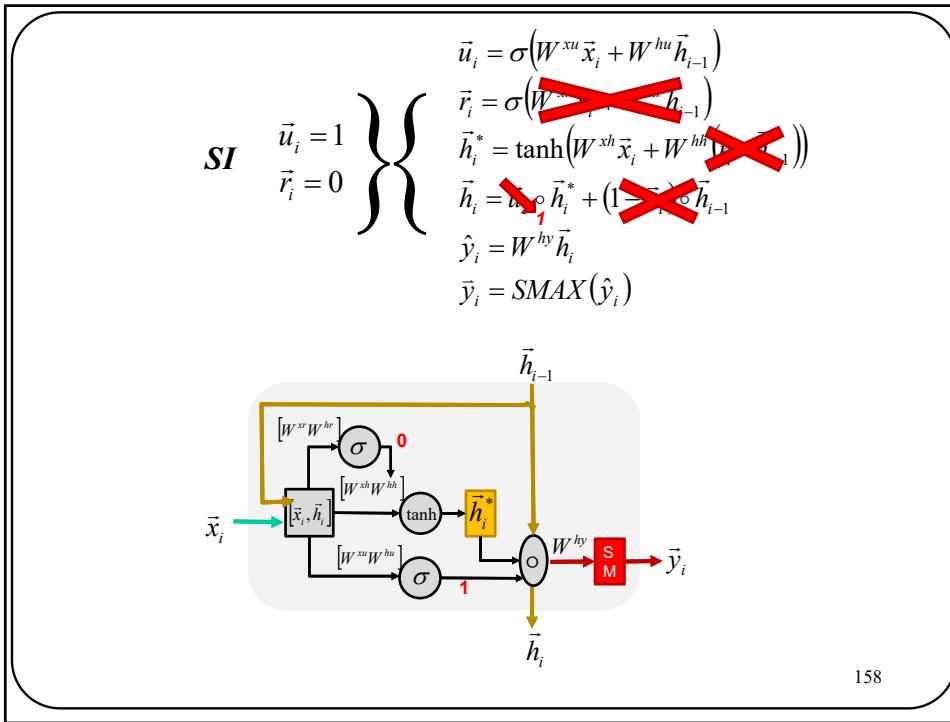
155



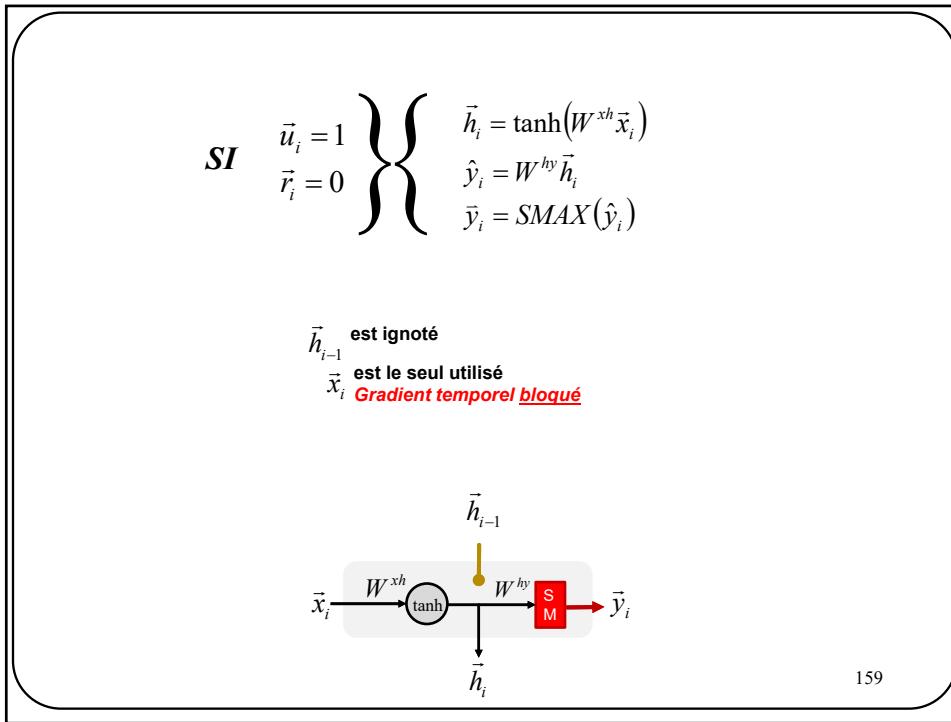
156



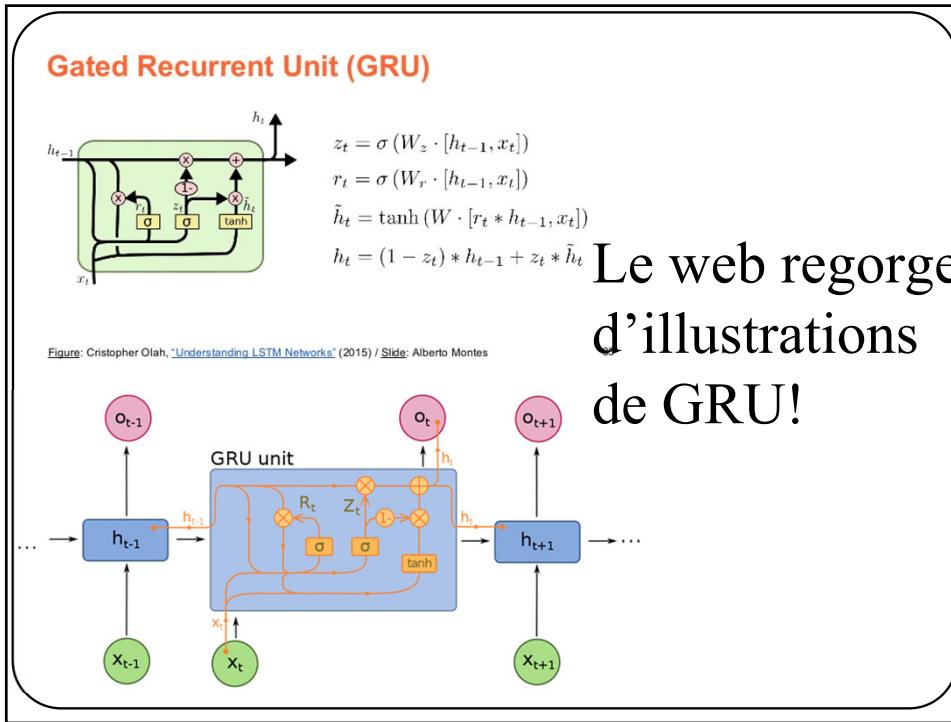
157



158

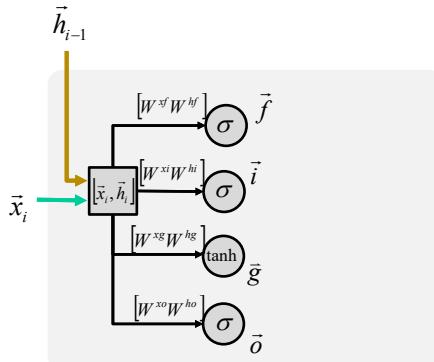


159



160

LSTM (Long Short Term Memory)

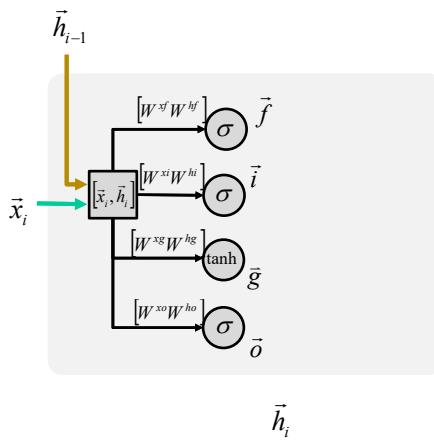


\vec{f} : forget gate
 \vec{i} : input gate
 \vec{g} : gate gate
 \vec{o} : output gate

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation, 1997

161

LSTM (Long Short Term Memory)



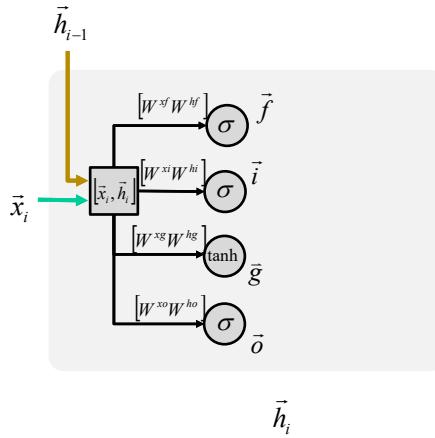
\vec{f} : forget gate
 \vec{i} : input gate
 \vec{g} : gate gate

Modèle récurrent
 le plus utilisé.
 À bien comprendre !

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation, 1997

162

LSTM (Long Short Term Memory)



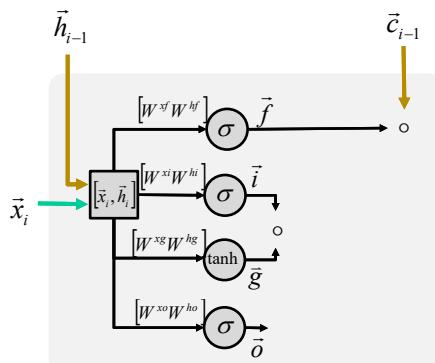
$$\begin{aligned}\vec{f} &= \sigma(W^{xf} \vec{x}_i + W^{hf} \vec{h}_{i-1}) \\ \vec{i} &= \sigma(W^{xi} \vec{x}_i + W^{hi} \vec{h}_{i-1}) \\ \bar{g} &= \tanh(W^{xg} \vec{x}_i + W^{hg} \vec{h}_{i-1}) \\ \vec{o} &= \sigma(W^{xo} \vec{x}_i + W^{ho} \vec{h}_{i-1})\end{aligned}$$

\vec{h}_i

163

163

LSTM (Long Short Term Memory)

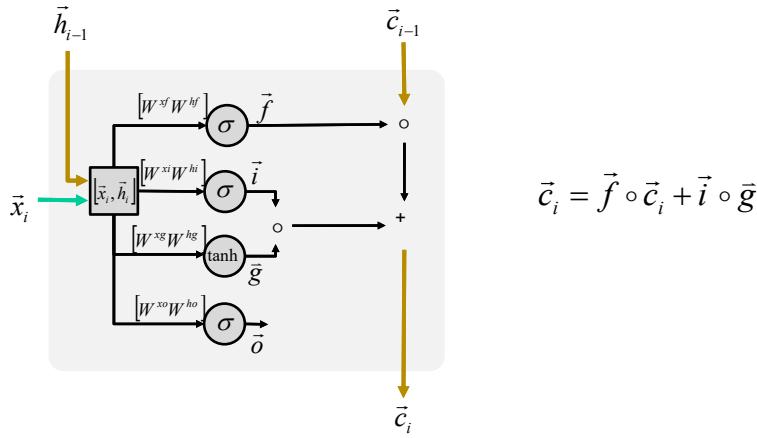


$$\begin{aligned}\vec{f} \circ \vec{c}_i \\ \vec{i} \circ \bar{g}\end{aligned}$$

164

164

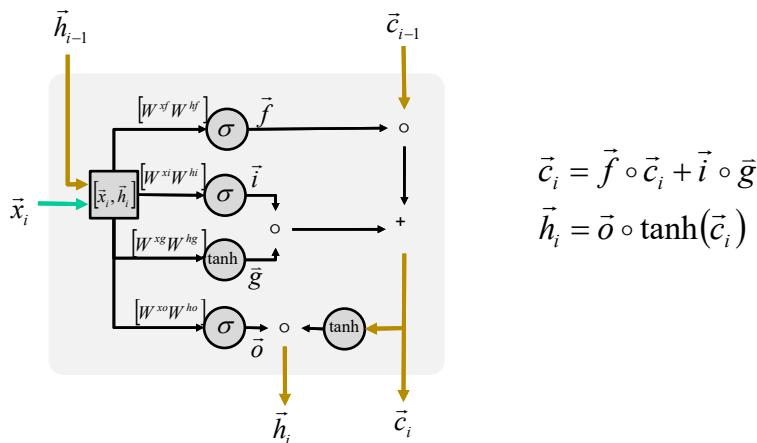
LSTM (Long Short Term Memory)



165

165

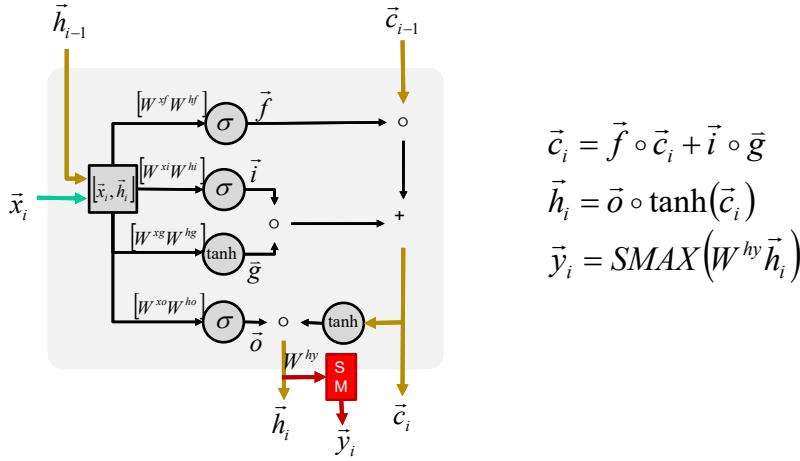
LSTM (Long Short Term Memory)



166

166

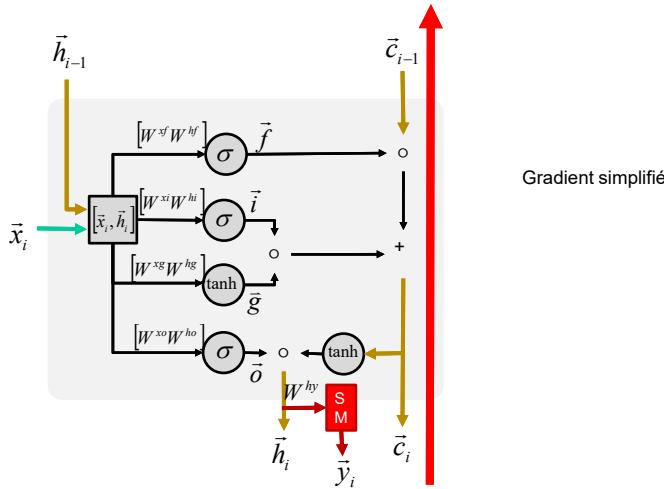
LSTM (Long Short Term Memory)



167

167

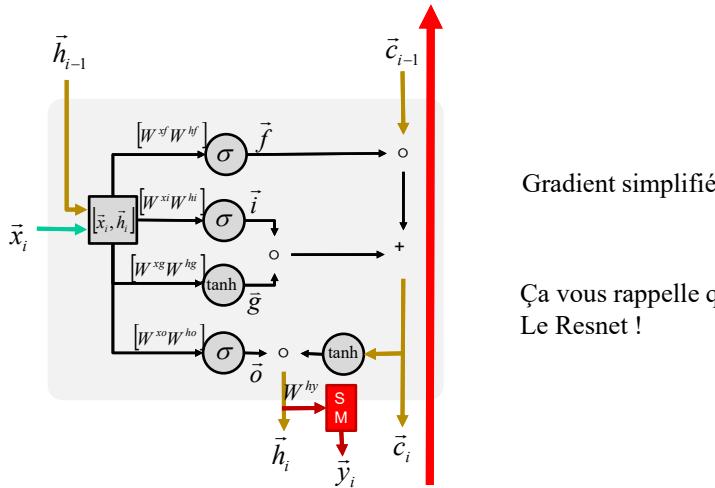
LSTM (Long Short Term Memory)



168

168

LSTM (Long Short Term Memory)



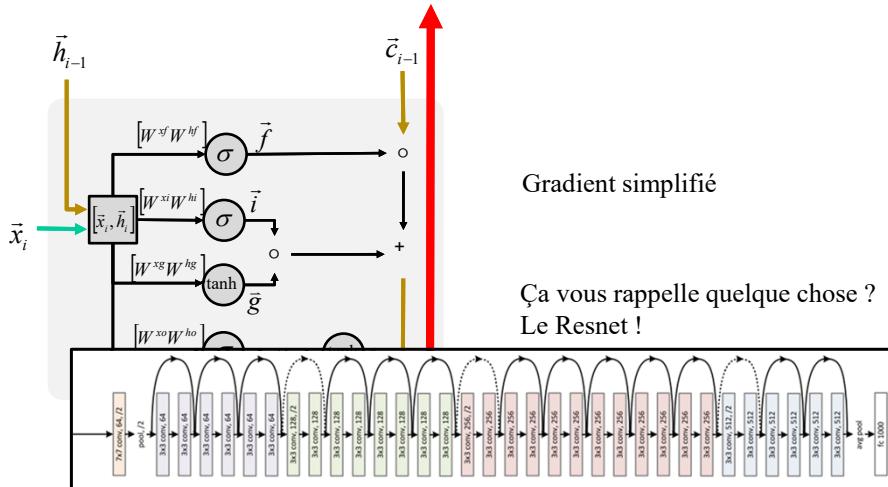
Gradient simplifié

Ça vous rappelle quelque chose ?
Le Resnet !

169

169

LSTM (Long Short Term Memory)



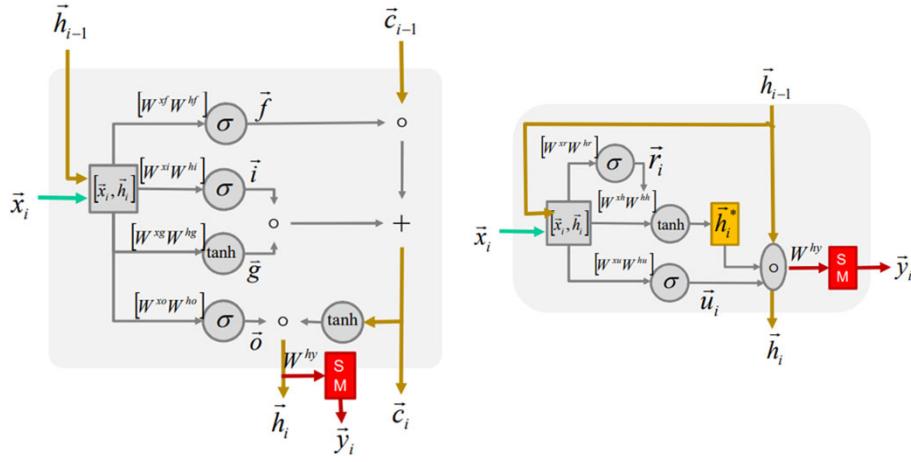
Gradient simplifié

Ça vous rappelle quelque chose ?
Le Resnet !

ResNET

170

LSTM et GRU

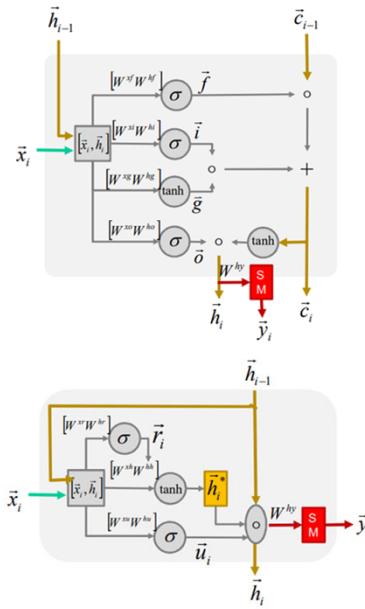


171

171

LSTM et GRU

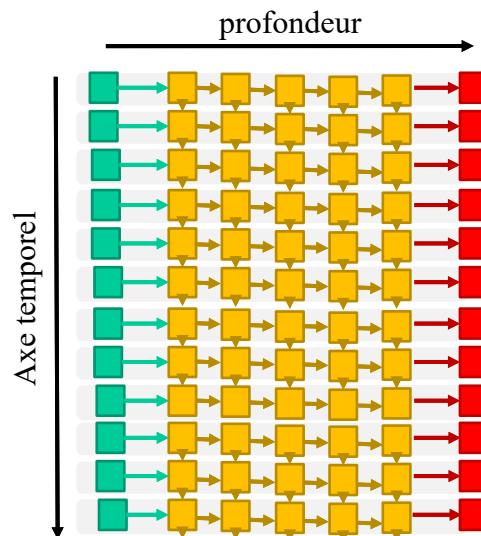
- Servent à protéger le gradient
- Conçus empiriquement
- GRU légèrement plus simple
- Les "gates" ne servent qu'à bloquer ou permettre à l'information (données ou temporelle) de passer



172

172

RNN multi-couches



173

Modèles d'attention

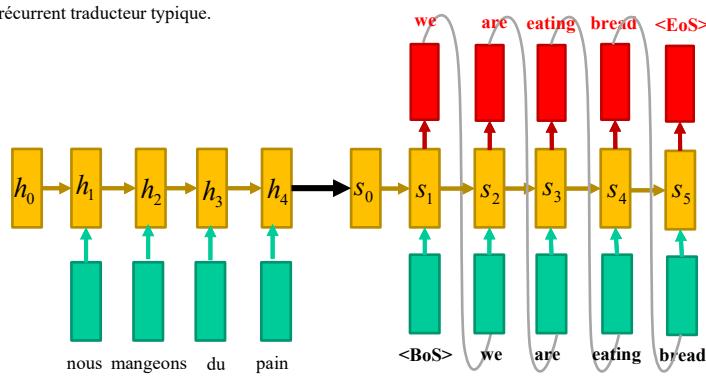
174

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’

Réseau récurrent traducteur typique.



175

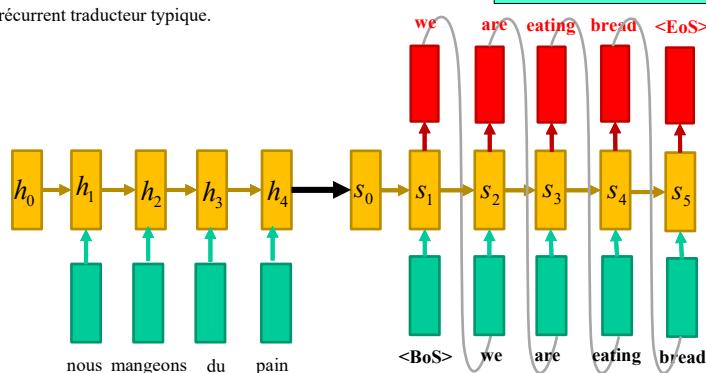
Seq2Seq:

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’

Problème avec cette formulation ?

Réseau récurrent traducteur typique.

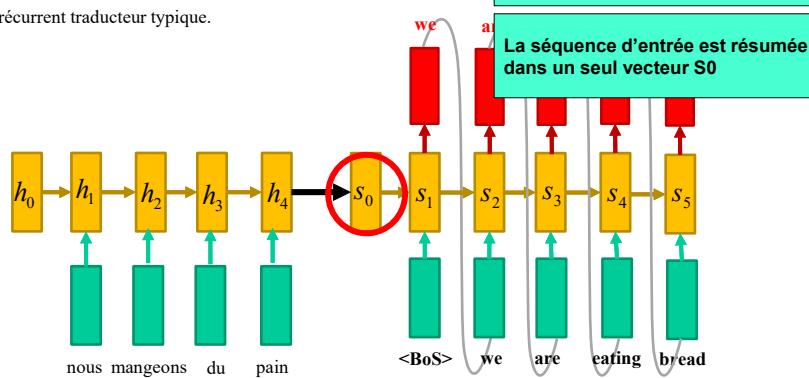


176

Seq2Seq:

'nous mangeons du pain' -> 'we are eating bread'

Réseau récurrent traducteur typique.



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

Problème avec cette formulation ?

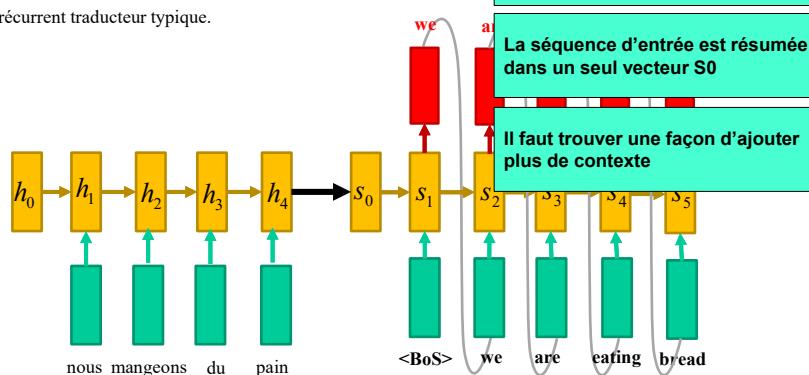
La séquence d'entrée est résumée dans un seul vecteur s_0

177

Seq2Seq:

'nous mangeons du pain' -> 'we are eating bread'

Réseau récurrent traducteur typique.



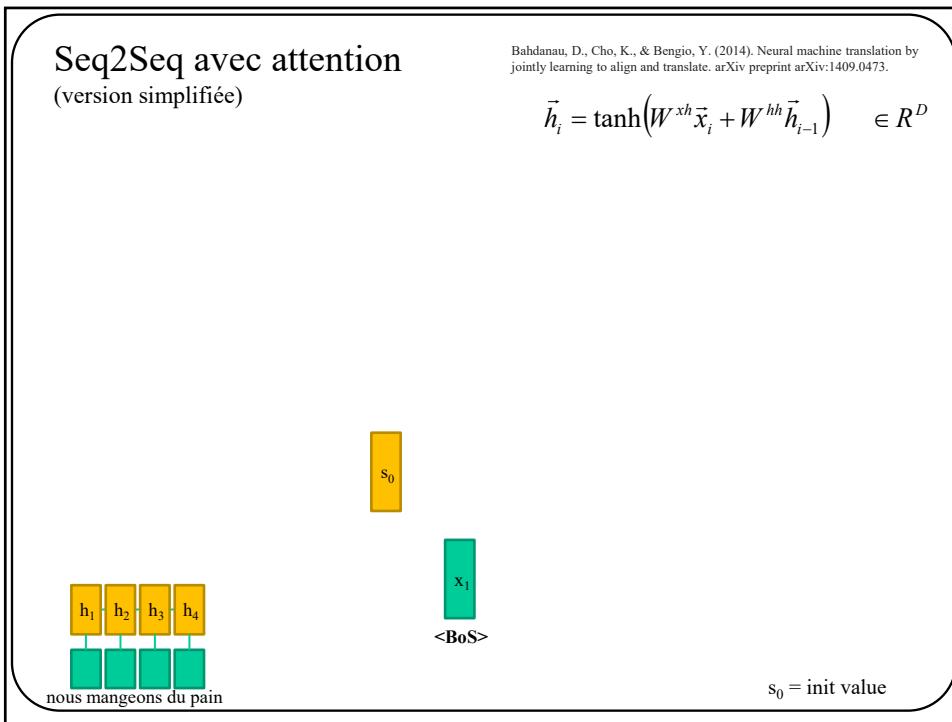
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Problème avec cette formulation ?

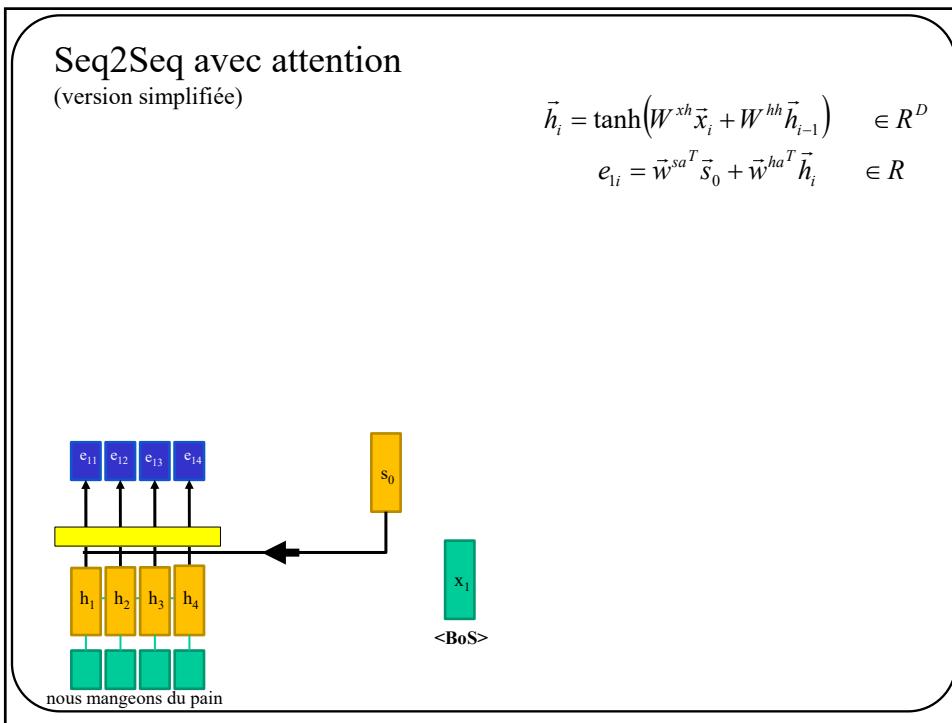
La séquence d'entrée est résumée dans un seul vecteur s_0

Il faut trouver une façon d'ajouter plus de contexte

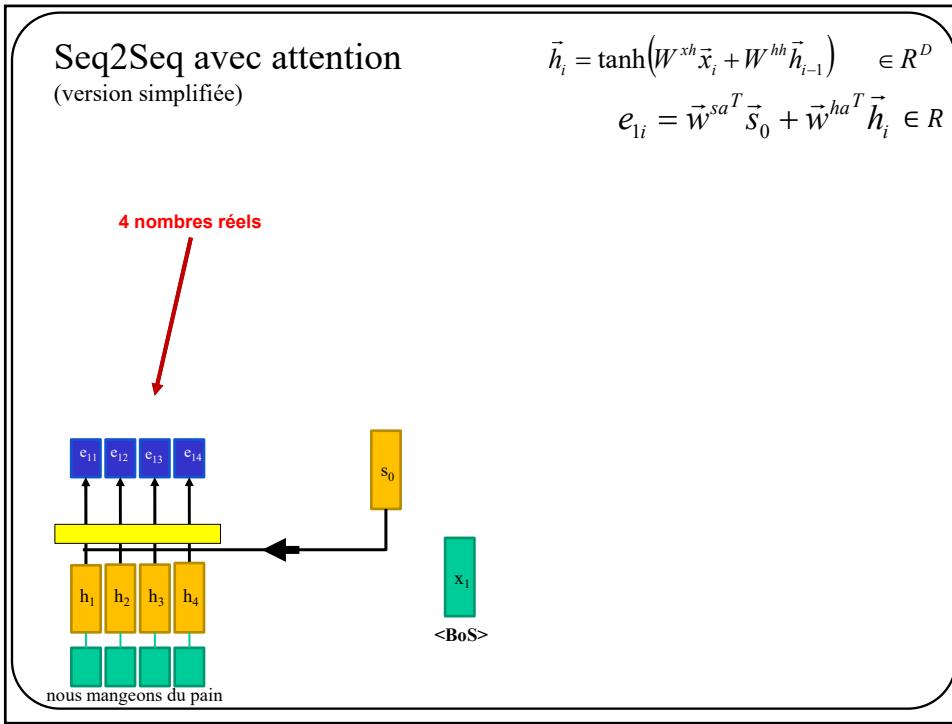
178



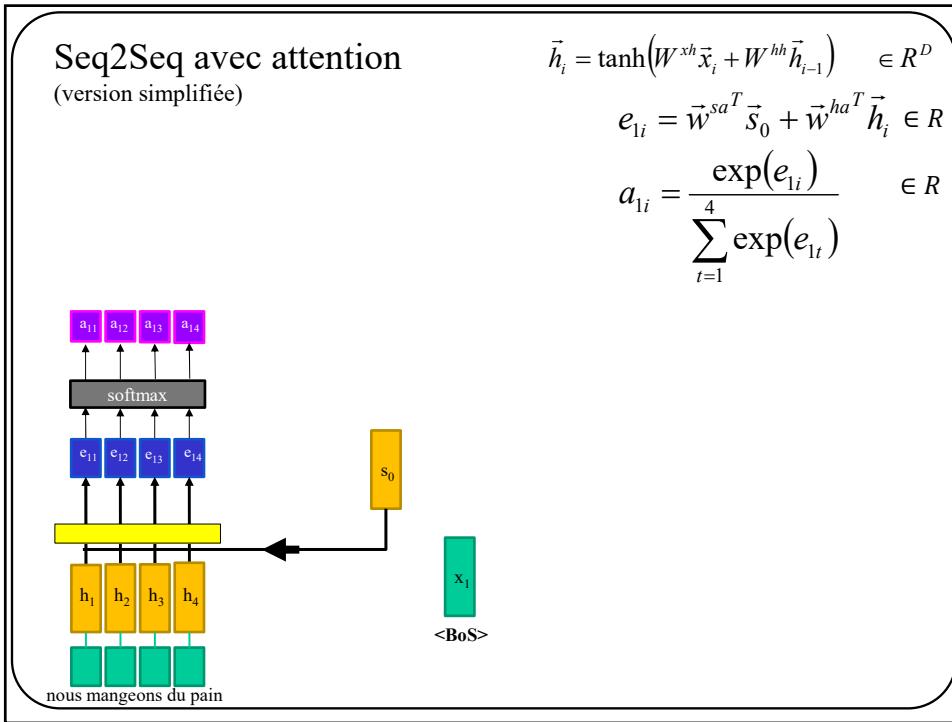
179



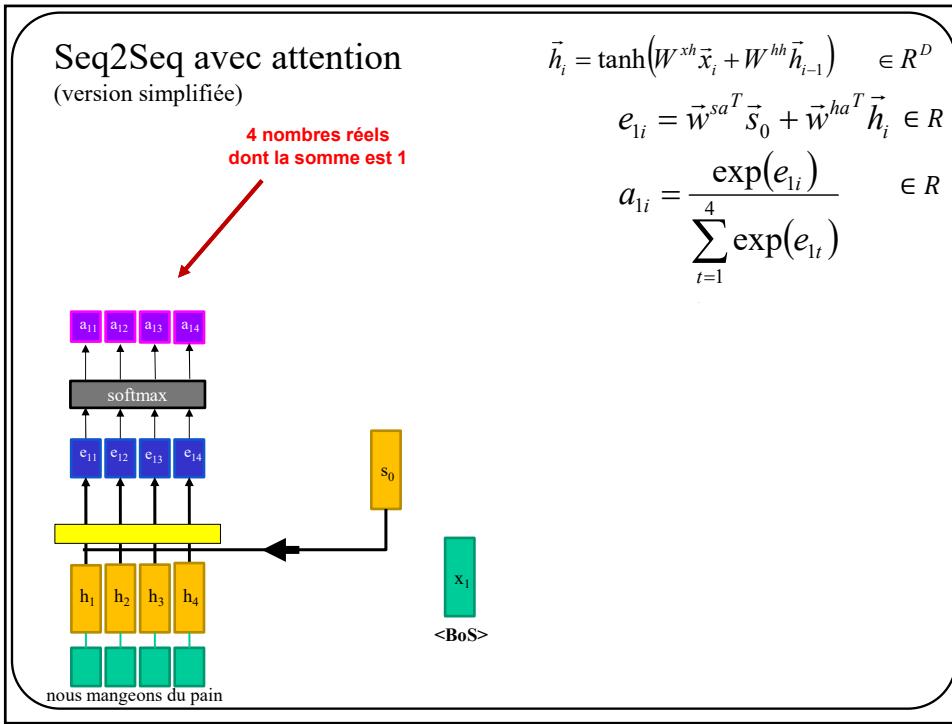
180



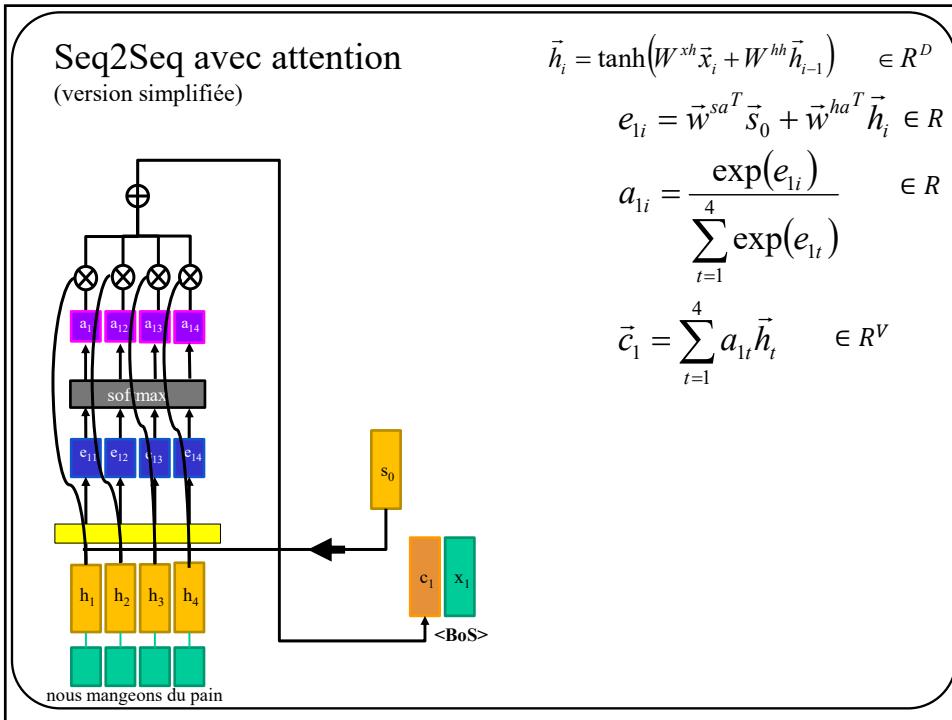
181



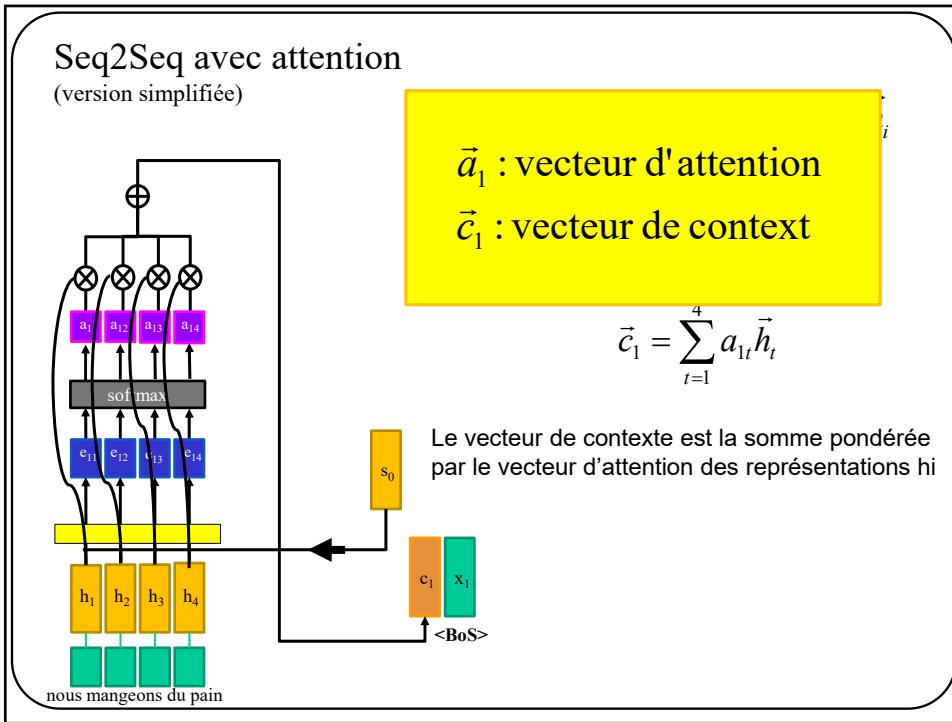
182



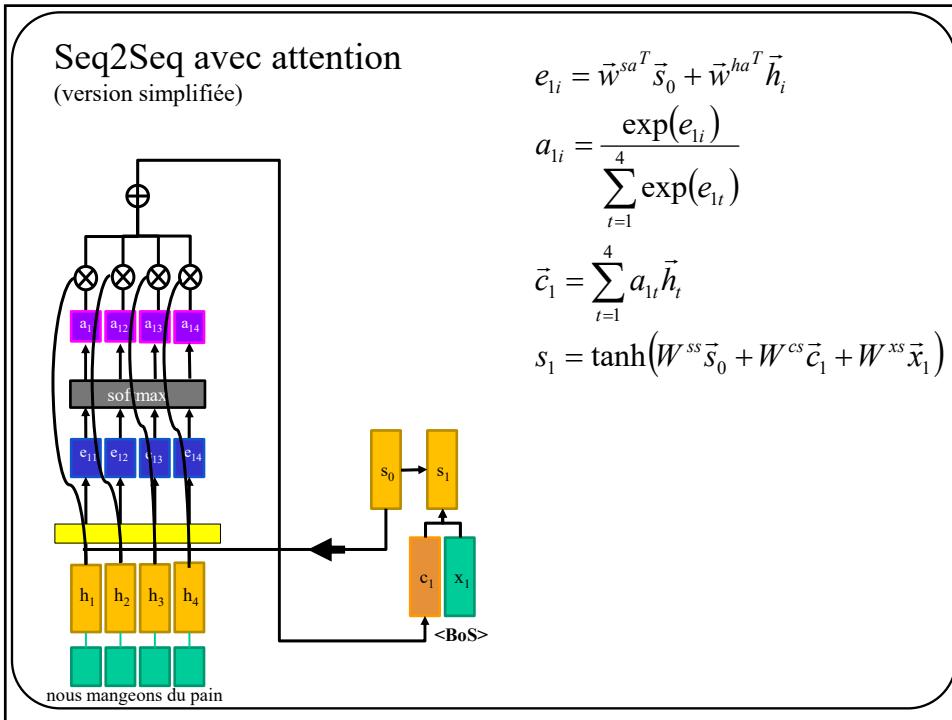
183



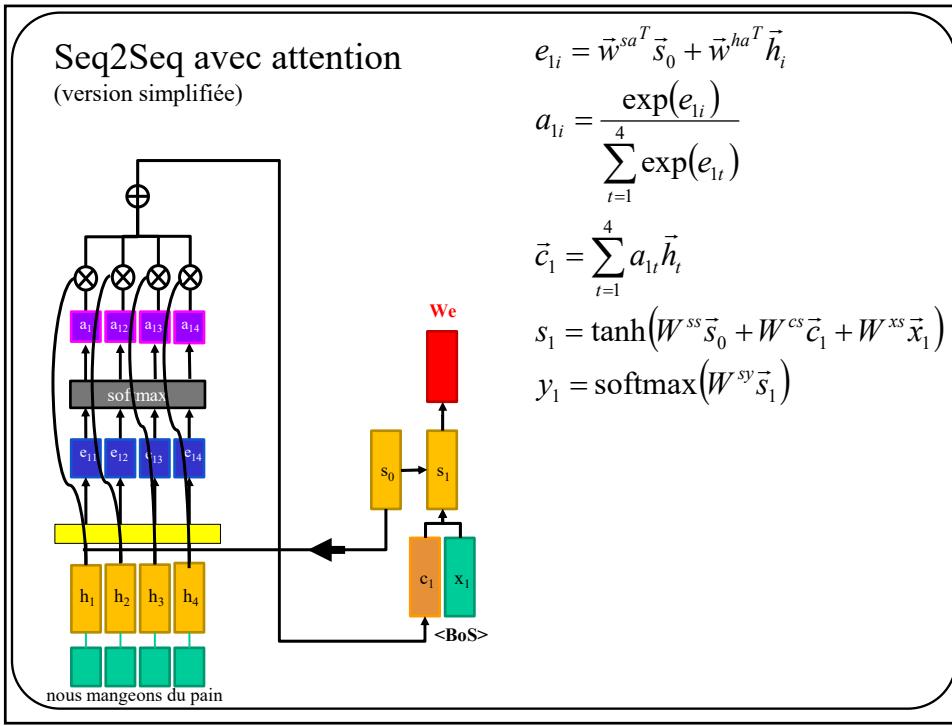
184



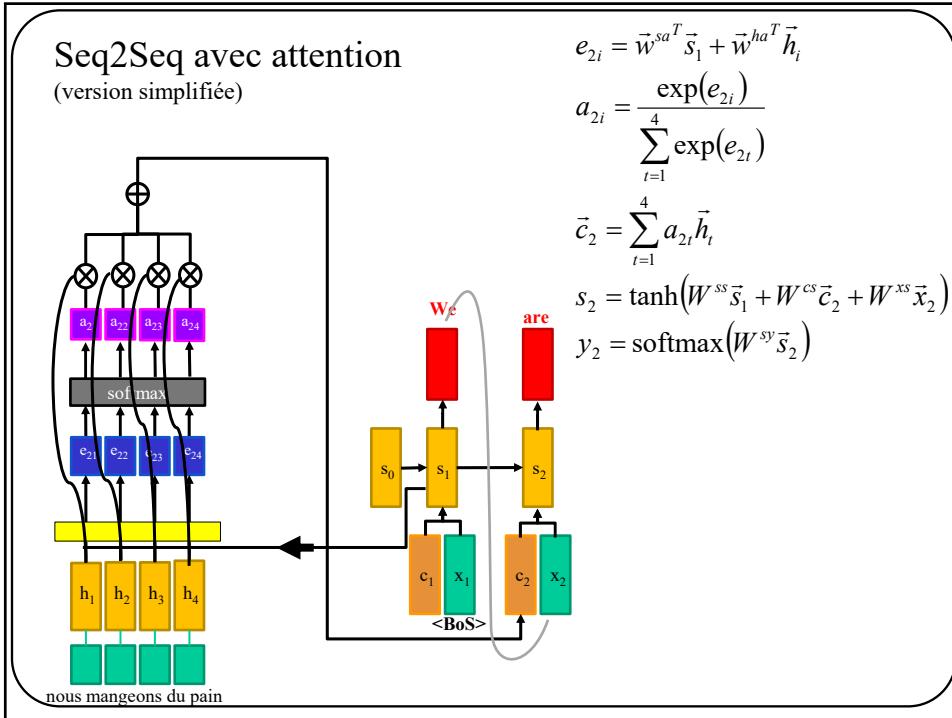
185



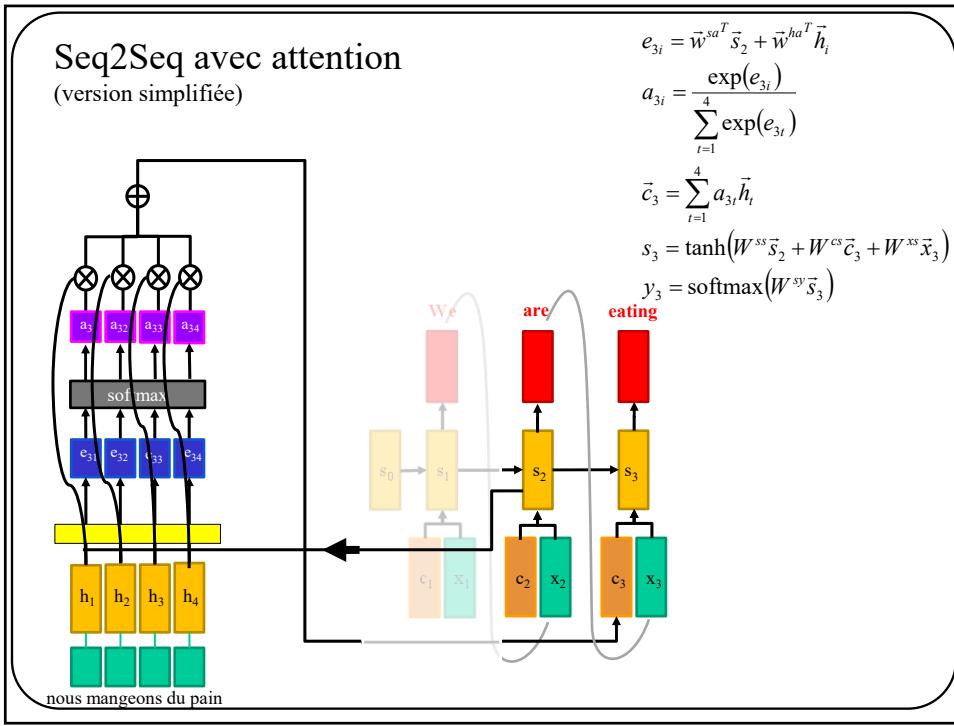
186



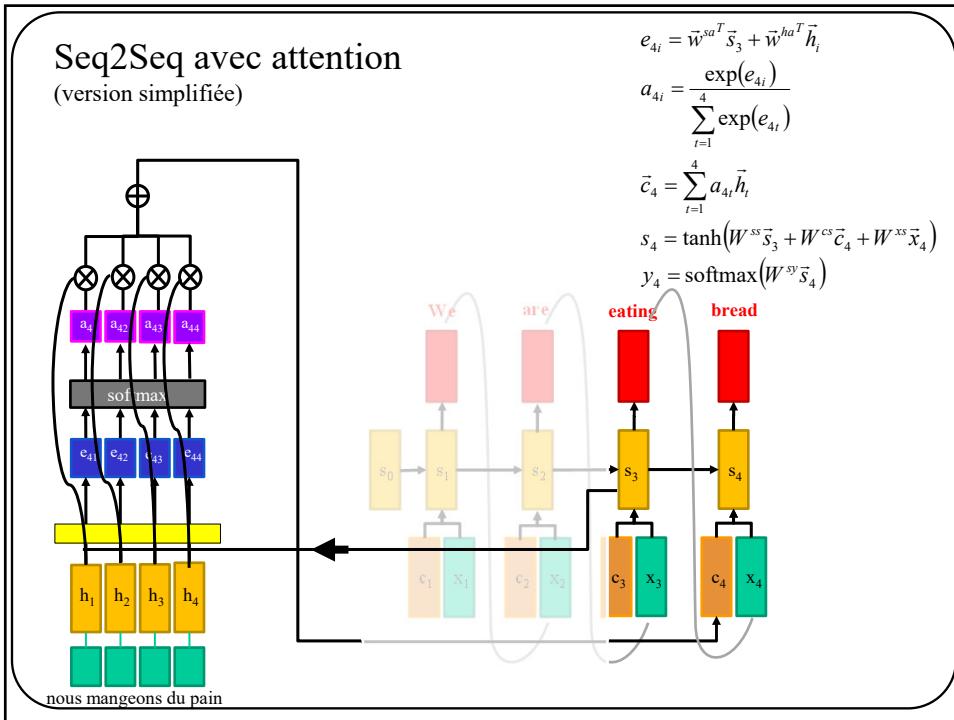
187



188

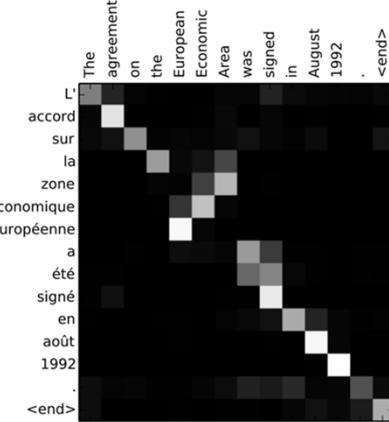


189

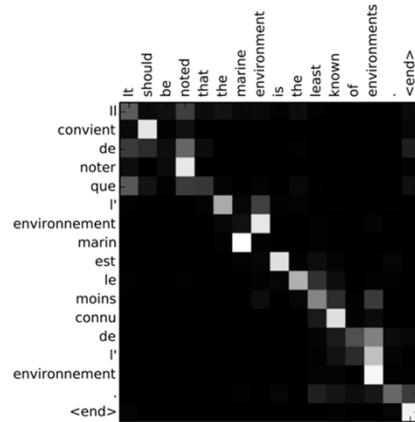


190

Un vecteur d'attention pour chaque mot



(a)

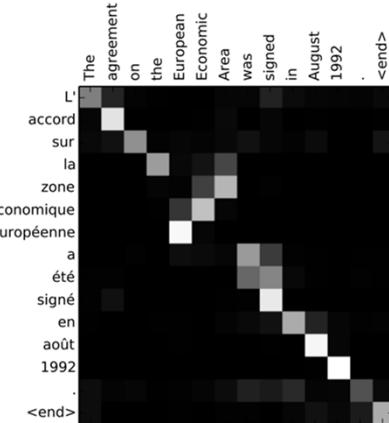


(b)

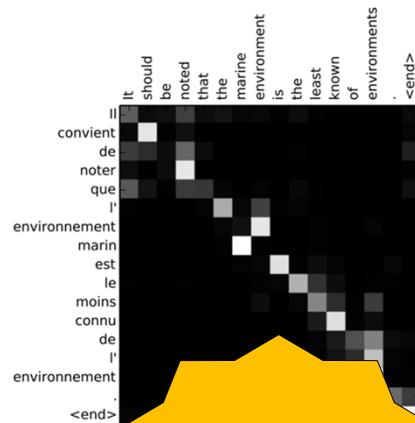
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

191

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 preprint arXiv:1409.0473.

192

L'auto-attention (*self attention*)

204

204

Revenons à la base : multiplication matricielle

Considérons les 4 matrices suivantes

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3 \times 4}$$

$$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^V = \begin{pmatrix} W^V_{11} & W^V_{12} & W^V_{13} \\ W^V_{21} & W^V_{22} & W^V_{23} \end{pmatrix} \in R^{2 \times 3}$$

205

205

Revenons à la base : multiplication matricielle

Leur multiplication donne:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3x4}$$

$$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3x3}$$

$$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3x3}$$

$$W^V = \begin{pmatrix} W^V_{11} & W^V_{12} & W^V_{13} \\ W^V_{21} & W^V_{22} & W^V_{23} \end{pmatrix} \in R^{2x3}$$

$$W^q X = Q = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \end{pmatrix} \in R^{3x4}$$

$$W^K X = K = \begin{pmatrix} K^x_{11} & K^x_{12} & K^x_{13} & K^x_{14} \\ K^x_{21} & K^x_{22} & K^x_{23} & K^x_{24} \\ K^x_{31} & K^x_{32} & K^x_{33} & K^x_{34} \end{pmatrix} \in R^{3x4}$$

$$W^V X = V = \begin{pmatrix} V_{11}^x & V_{12}^x & V_{13}^x & V_{14}^x \\ V_{21}^x & V_{22}^x & V_{23}^x & V_{24}^x \end{pmatrix} \in R^{2x4}$$

206

206

Auto attention

X est une matrice de données pour laquelle chaque colonne i correspond au jeton d'une mot \vec{x}_i

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3x4}$$

Dans cet exemple, 4 mots en entrée donc 4 colonnes dans X
Les jetons peuvent être obtenus par **Word2Vec**

207

207

Auto attention

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3 \times 4}$$

$$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^v = \begin{pmatrix} W^v_{11} & W^v_{12} & W^v_{13} \\ W^v_{21} & W^v_{22} & W^v_{23} \end{pmatrix} \in R^{2 \times 3}$$

W : Matrices de paramètres appris par **rétropropagation**

208

208

Auto attention

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3 \times 4}$$

$$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^v = \begin{pmatrix} W^v_{11} & W^v_{12} & W^v_{13} \\ W^v_{21} & W^v_{22} & W^v_{23} \end{pmatrix} \in R^{2 \times 3}$$

Matrices de paramètres appris par rétropropagation

Pour ces 3 matrices, le nombre de colonnes (3) doit être égale au nombre de lignes dans X (3)

209

209

Auto attention

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3 \times 4}$$

Nous mangeons du pain

$$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3 \times 3}$$

$$W^v = \begin{pmatrix} W^v_{11} & W^v_{12} & W^v_{13} \\ W^v_{21} & W^v_{22} & W^v_{23} \end{pmatrix} \in R^{2 \times 3}$$

Matrices de paramètres appris par rétropropagation
Pour ces 3 matrices, le nombre de ligne (3,3,2) est arbitraire

210

210

Q,K,V contiennent une transformation linéaire de la matrice d'entrée X. Chaque jeton **Xi** a été transformé en des **vecteurs Qi, Ki et Vi**

$$W^V X$$

$$K : \begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \end{pmatrix}$$

$$W^K X$$

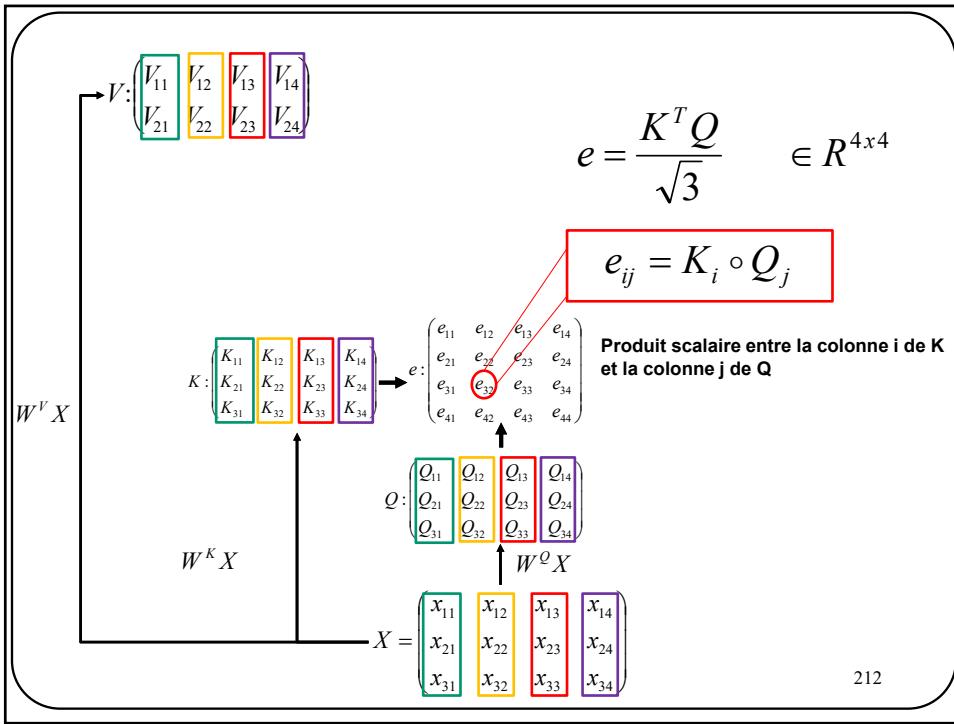
$$Q : \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \end{pmatrix}$$

$$W^Q X$$

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix}$$

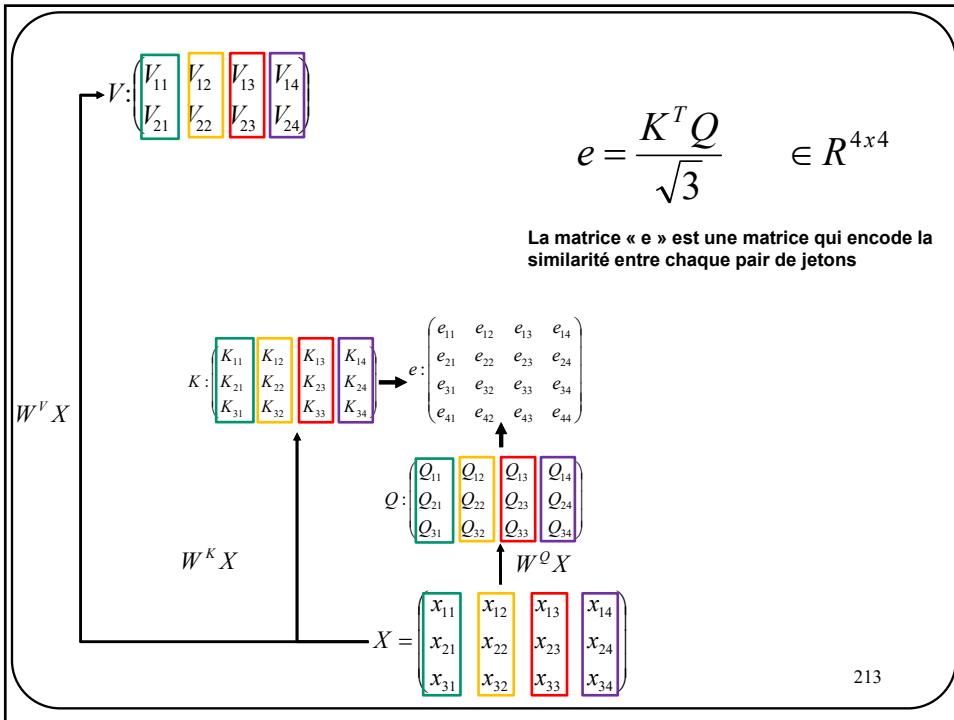
211

211



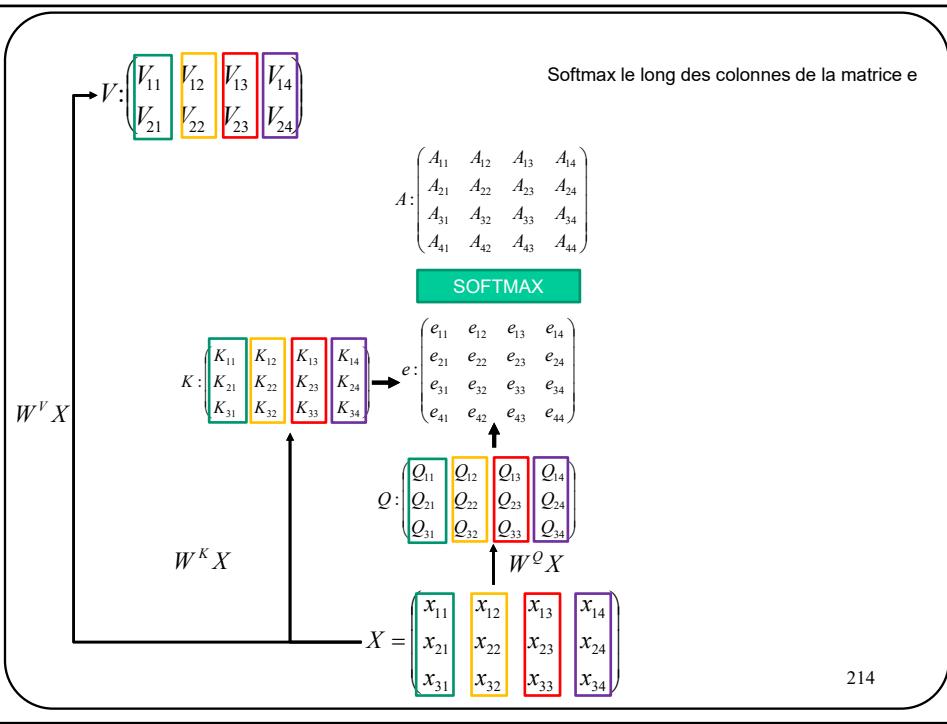
212

212

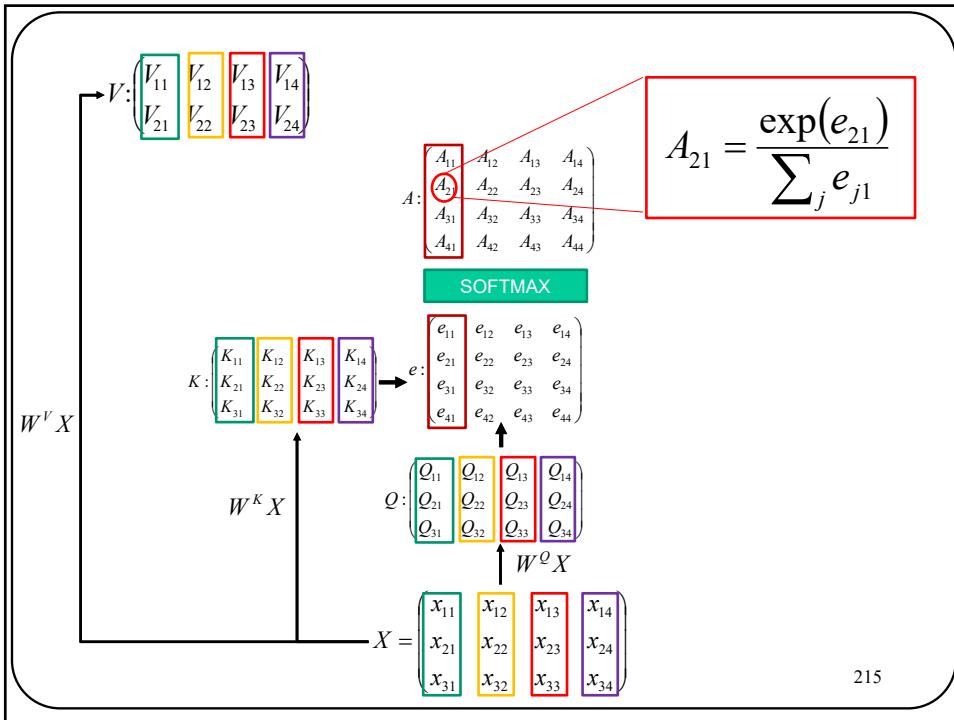


213

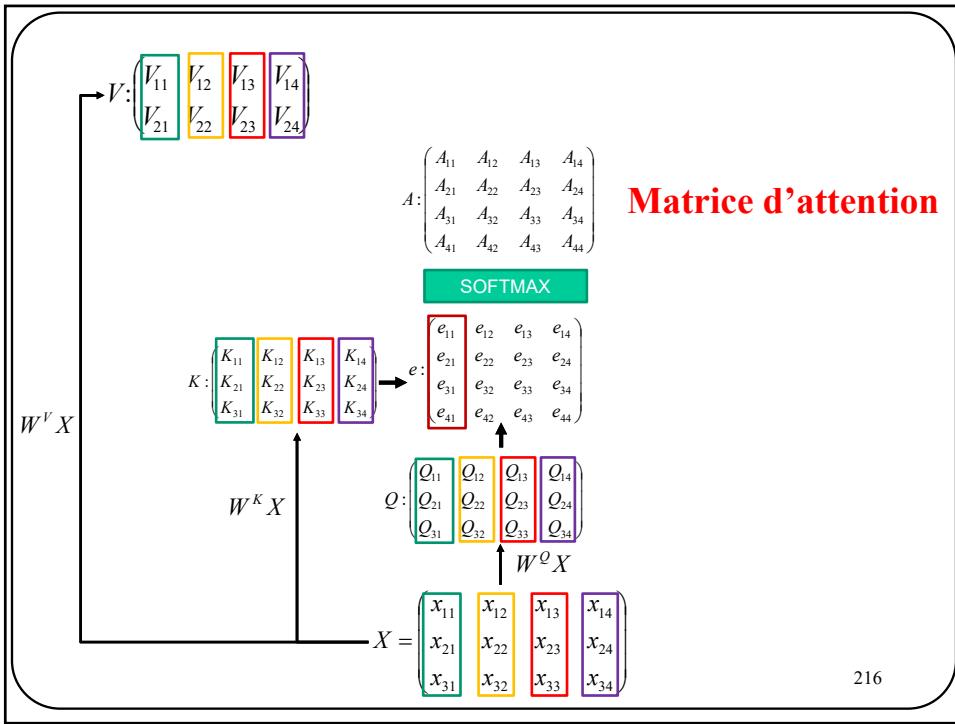
213



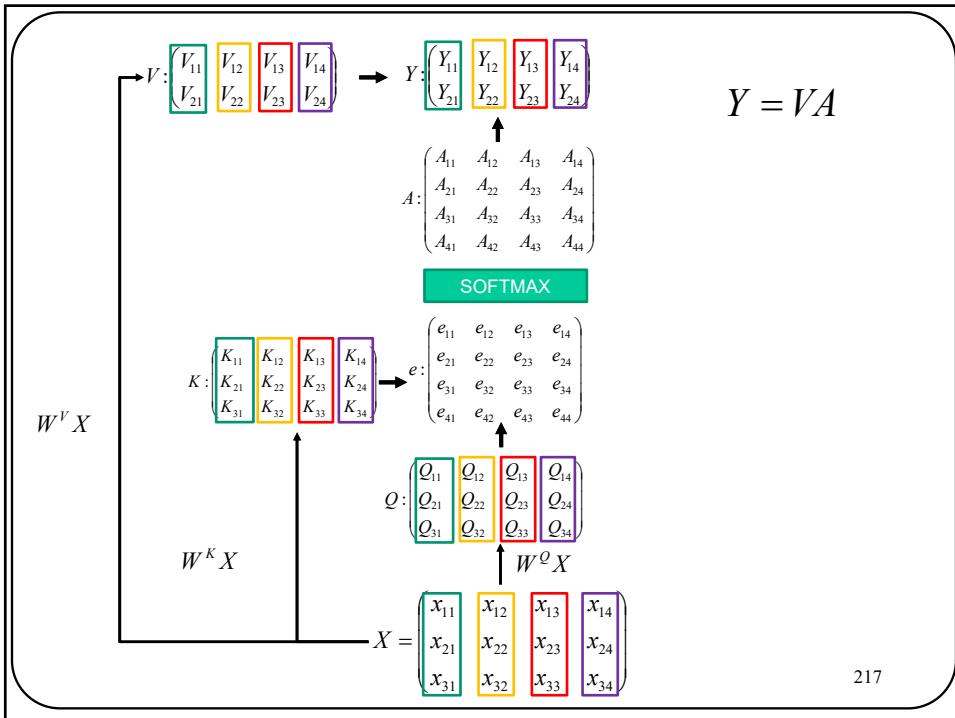
214



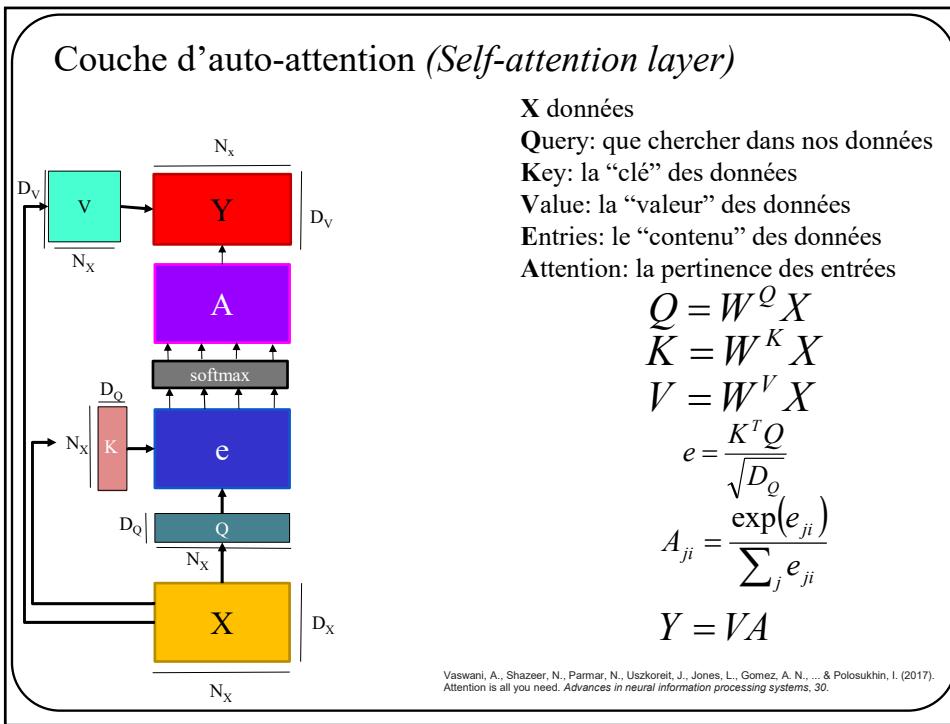
215



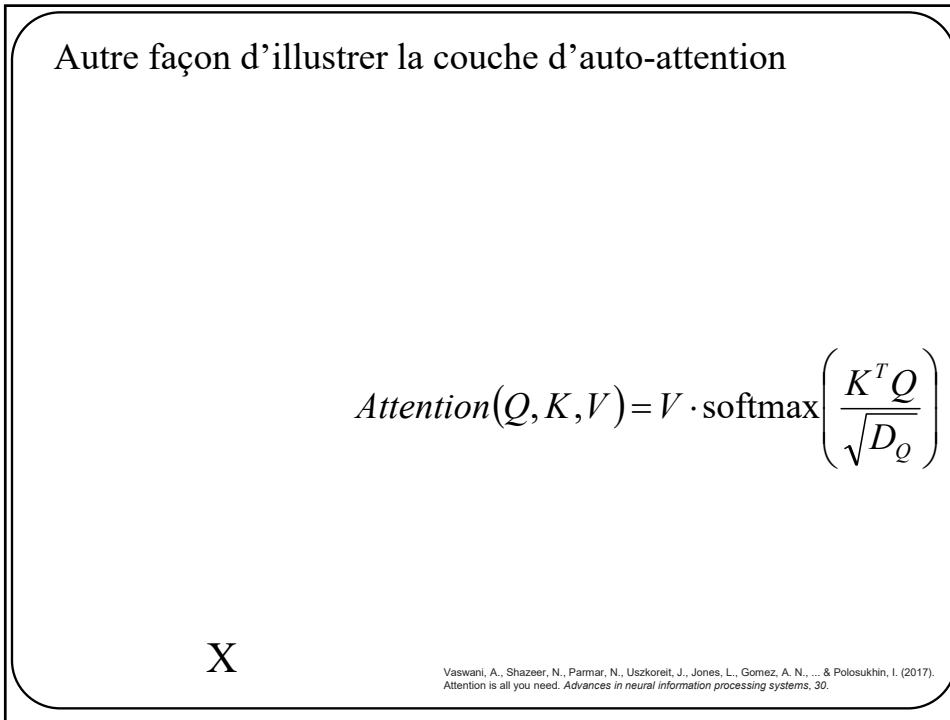
216



217



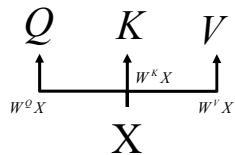
218



219

Autre façon d'illustrer la couche d'auto-attention

$$\text{Attention}(Q, K, V) = V \cdot \text{softmax} \left(\frac{K^T Q}{\sqrt{D_Q}} \right)$$

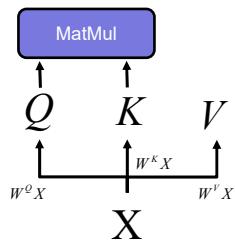


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

220

Autre façon d'illustrer la couche d'auto-attention

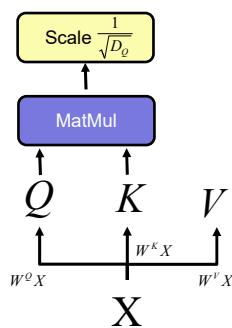
$$\text{Attention}(Q, K, V) = V \cdot \text{softmax} \left(\frac{K^T Q}{\sqrt{D_Q}} \right)$$



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.

221

Autre façon d'illustrer la couche d'auto-attention

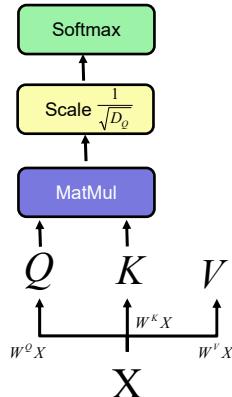


$$\text{Attention}(Q, K, V) = V \cdot \text{softmax} \left(\frac{K^T Q}{\sqrt{D_Q}} \right)$$

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.

222

Autre façon d'illustrer la couche d'auto-attention

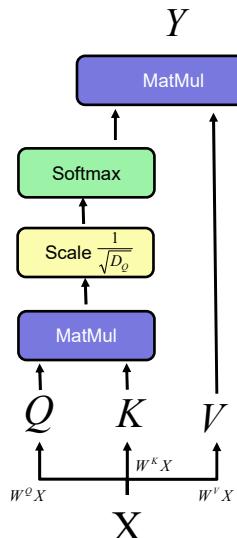


$$\text{Attention}(Q, K, V) = V \cdot \text{softmax} \left(\frac{K^T Q}{\sqrt{D_Q}} \right)$$

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.

223

Autre façon d'illustrer la couche d'auto-attention

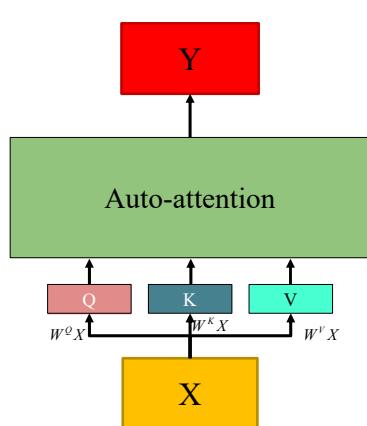


$$\text{Attention}(Q, K, V) = V \cdot \text{softmax}\left(\frac{K^T Q}{\sqrt{D_Q}}\right)$$

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.

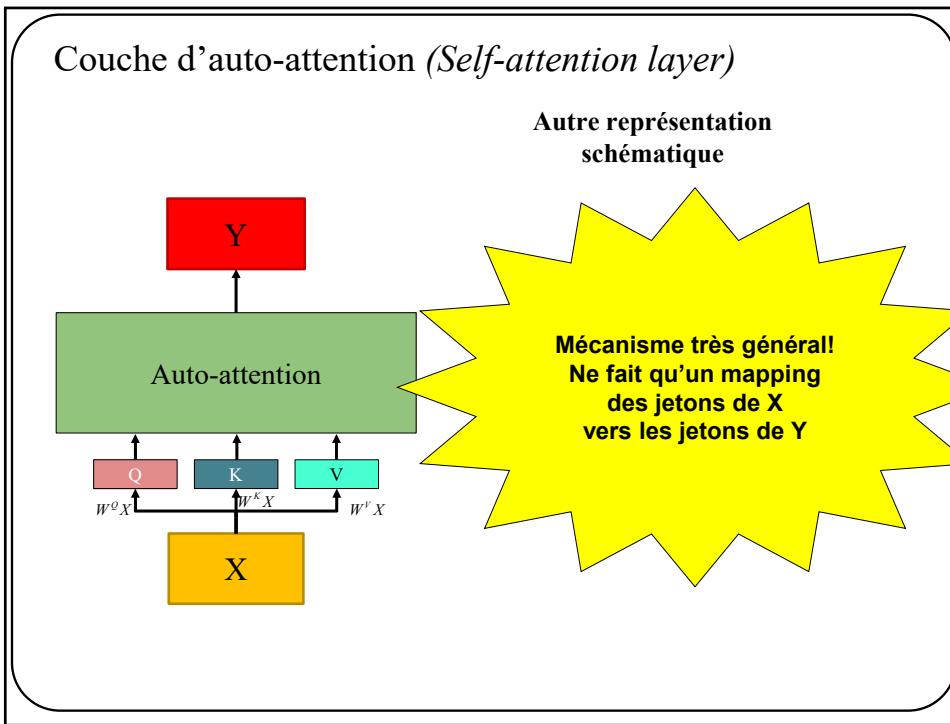
224

Couche d'auto-attention (*Self-attention layer*)

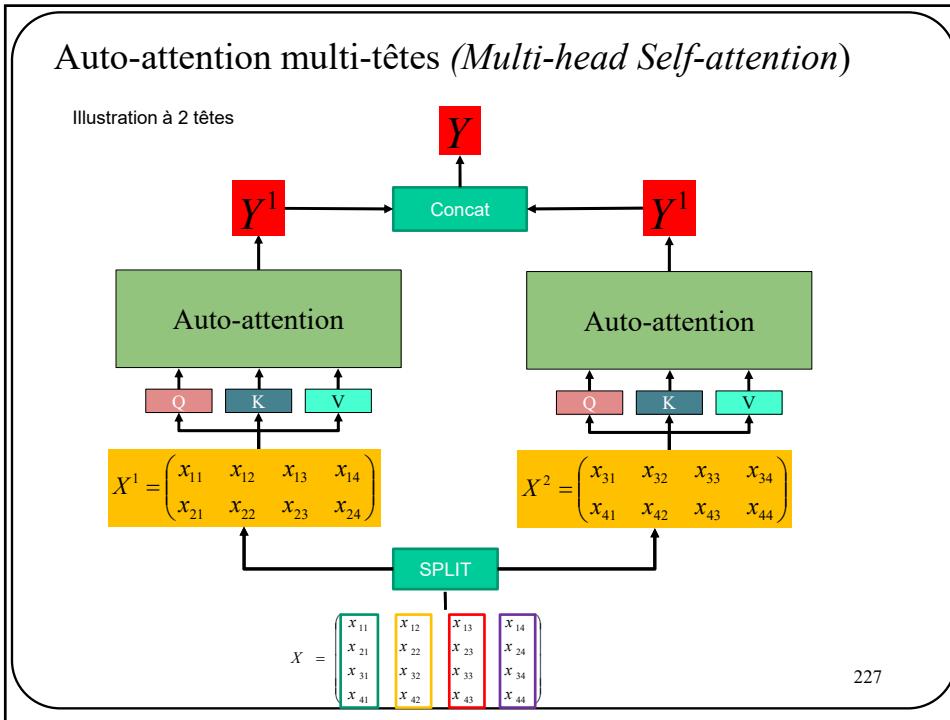


Autre représentation schématique

225

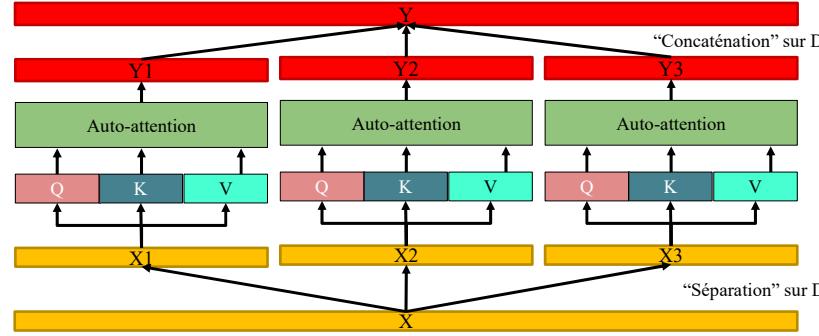


226



227

Auto-attention multi-tête (*Multi-head Self-attention*)



Très bon résumé de l'auto-attention multi-tête :

<https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853>

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.

228

L'apothéose des réseaux de neurones

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.

229

Transformer

Implique **aucune notion de récurrence**

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.

230

Transformer (Attention is all you need)

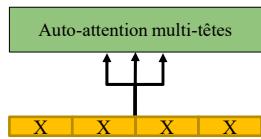
X | X | X | X

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.

231

Transformer (*Attention is all you need*)

- Auto-attention multi-têtes sur les dimensions de X

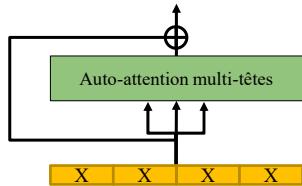


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

232

Transformer (*Attention is all you need*)

- Auto-attention multi-têtes sur les dimensions de X
- “+” = connexion résiduelle

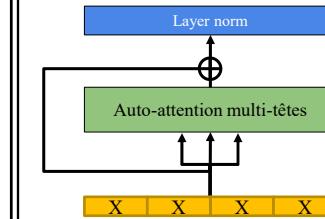


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

233

Transformer (*Attention is all you need*)

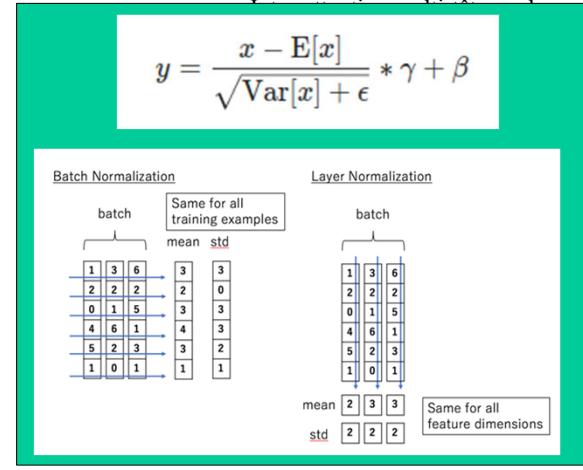
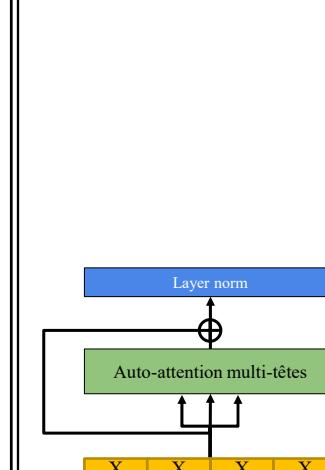
- Auto-attention multi-têtes sur les dimensions de X
- “+” = connexion résiduelle
- “Layer-norm”



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.

234

Transformer (*Attention is all you need*)



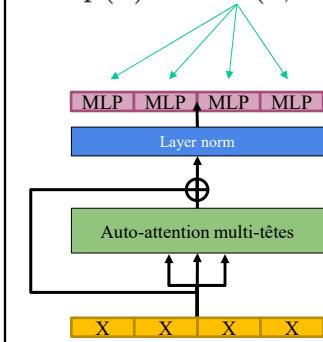
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

235

Transformer (*Attention is all you need*)

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

- Auto-attention multi-têtes sur les dimensions de X
- “+” = connexion résiduelle
- “Layer-norm”
- MLP pour chaque jeton

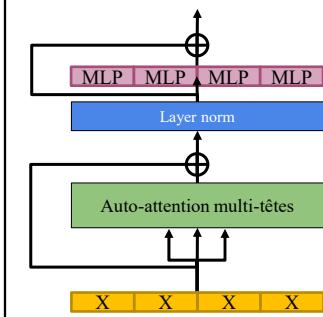


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

236

Transformer (*Attention is all you need*)

- Intra-attention multi-tête sur les dimensions de X
- “+” = connexion résiduelle
- “Layer-norm”
- MLP pour chaque jeton
- 2e “+” = connexion résiduelle

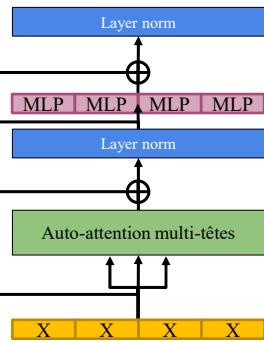


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

237

Transformer (*Attention is all you need*)

- Auto-attention multi-têtes sur les dimensions de X
- “+” = connexion résiduelle
- “Layer-norm”
- MLP pour chaque jeton
- 2e “+” = connexion résiduelle
- 2e “Layer-norm”

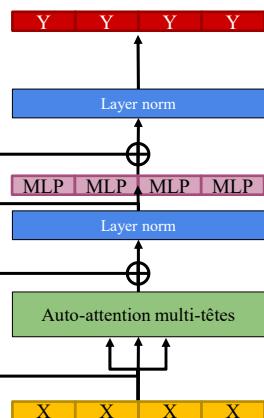


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

238

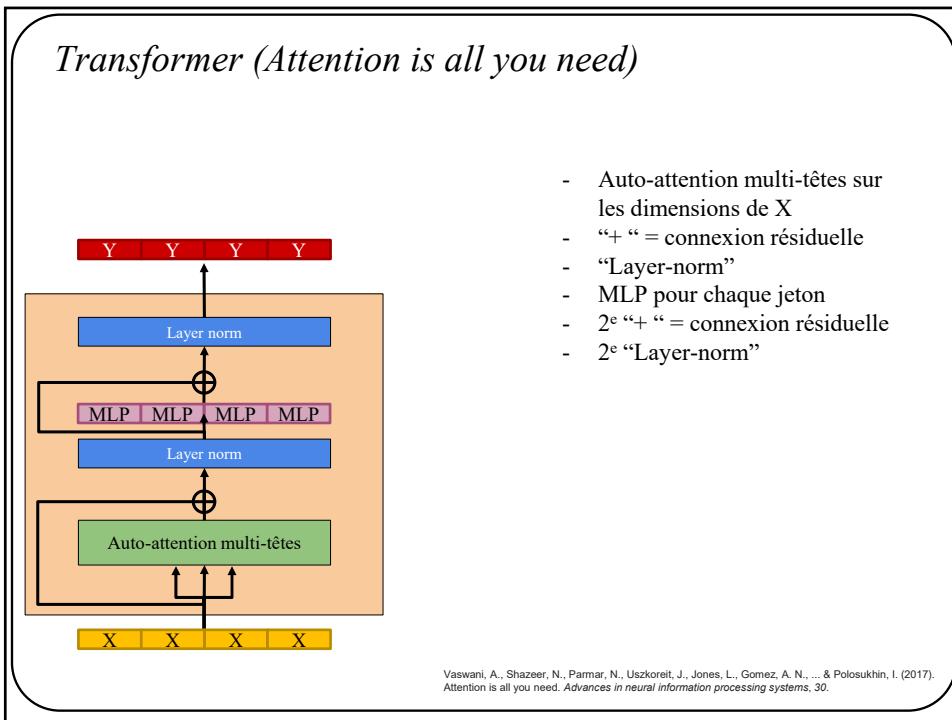
Transformer (*Attention is all you need*)

- Auto-attention multi-têtes sur les dimensions de X
- “+” = connexion résiduelle
- “Layer-norm”
- MLP pour chaque jeton
- 2e “+” = connexion résiduelle
- 2e “Layer-norm”

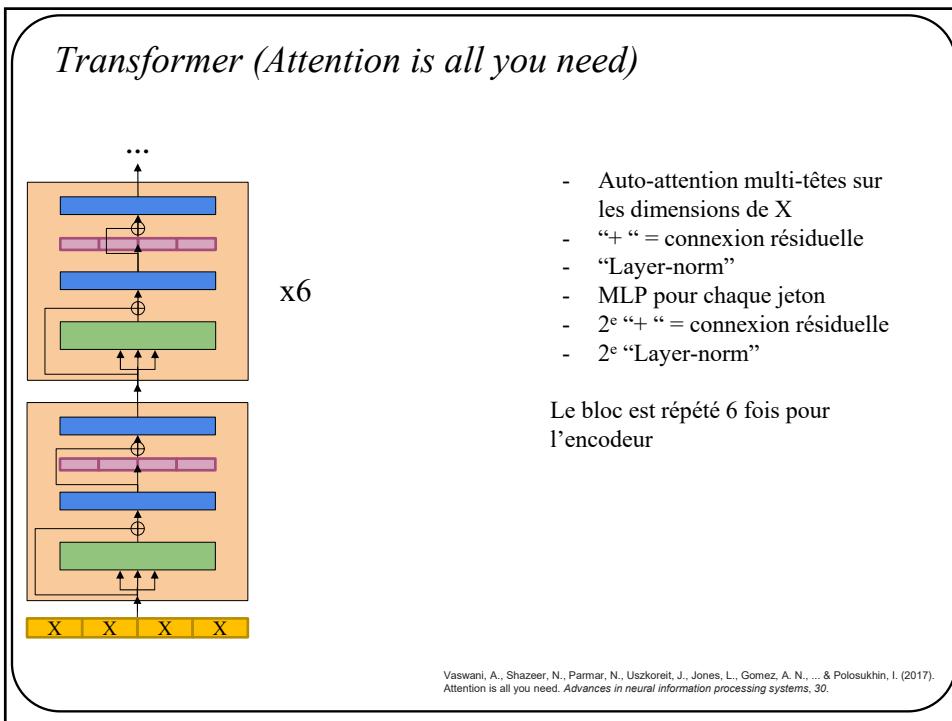


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

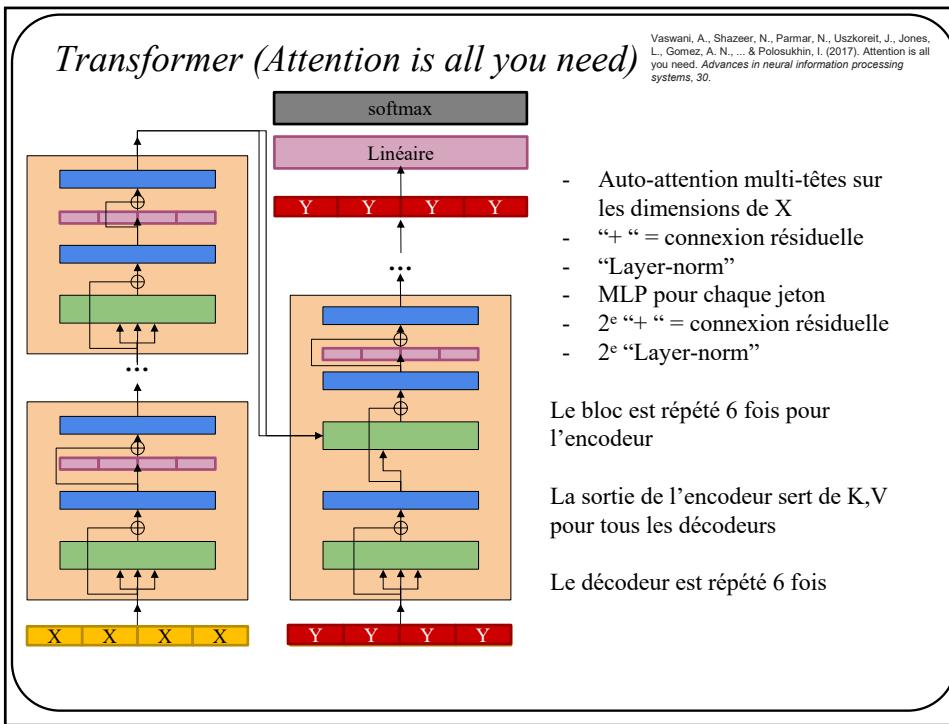
239



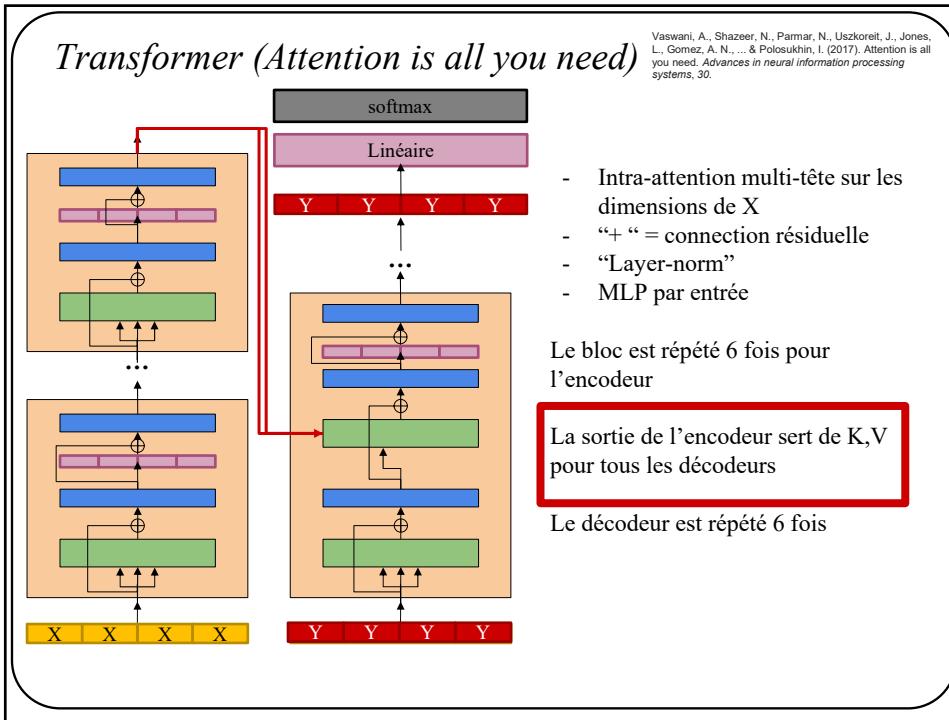
240



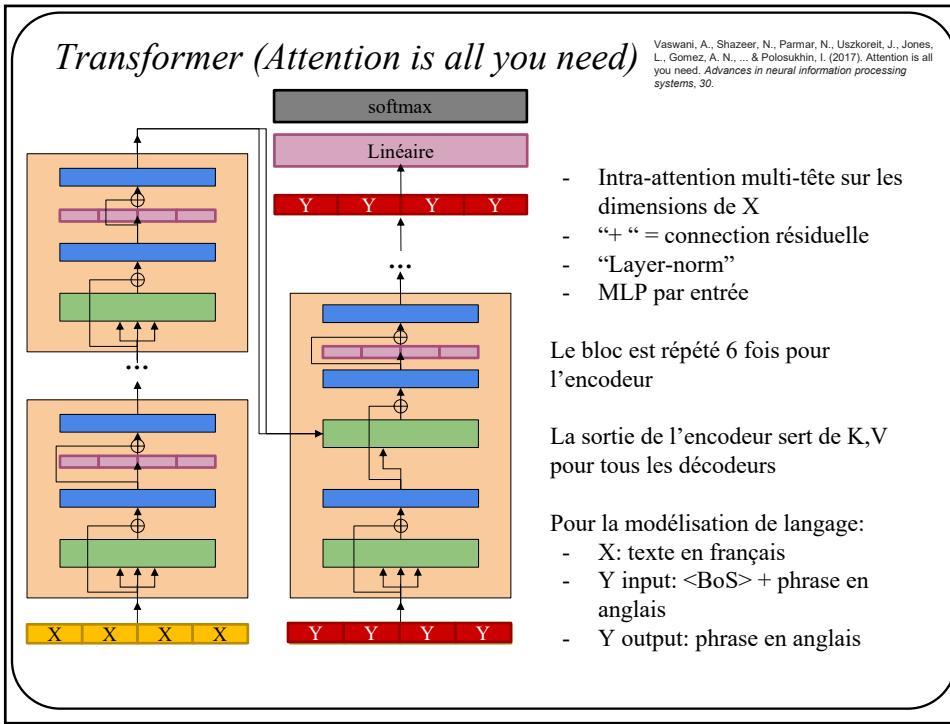
241



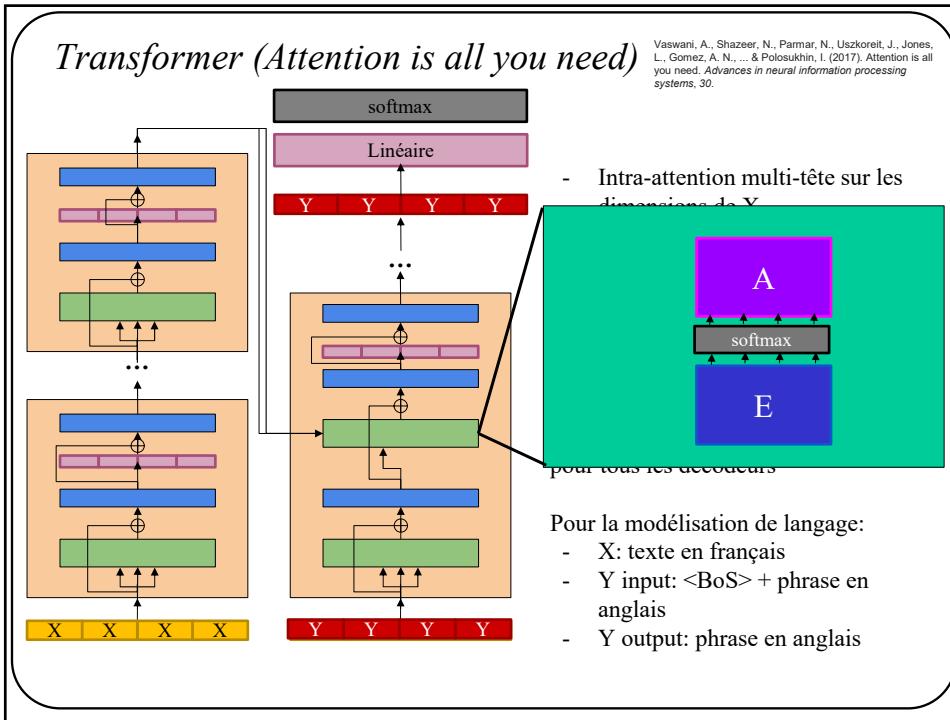
242



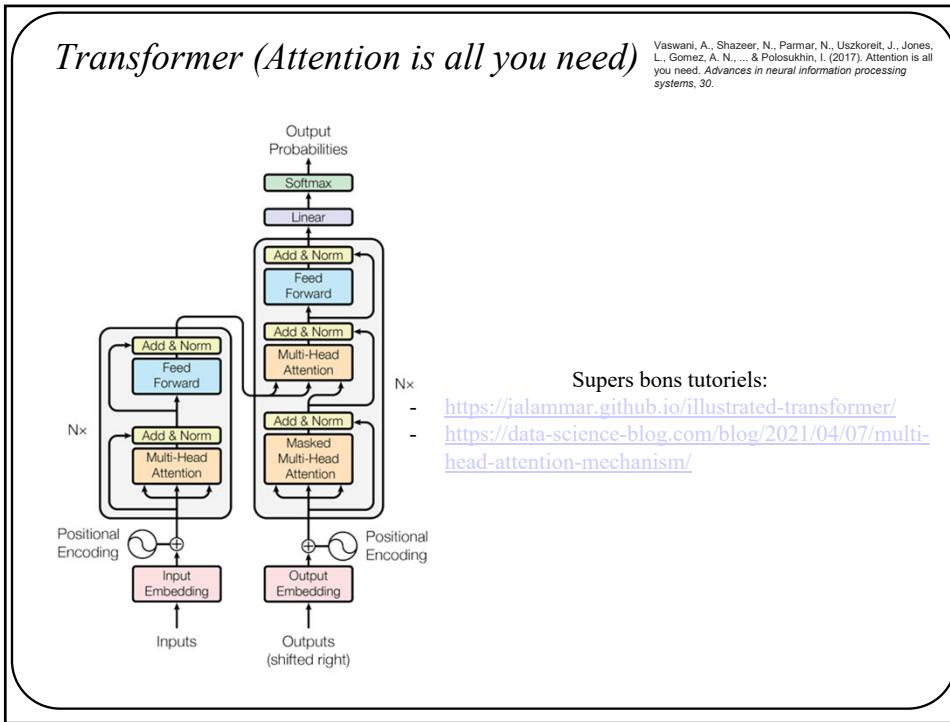
243



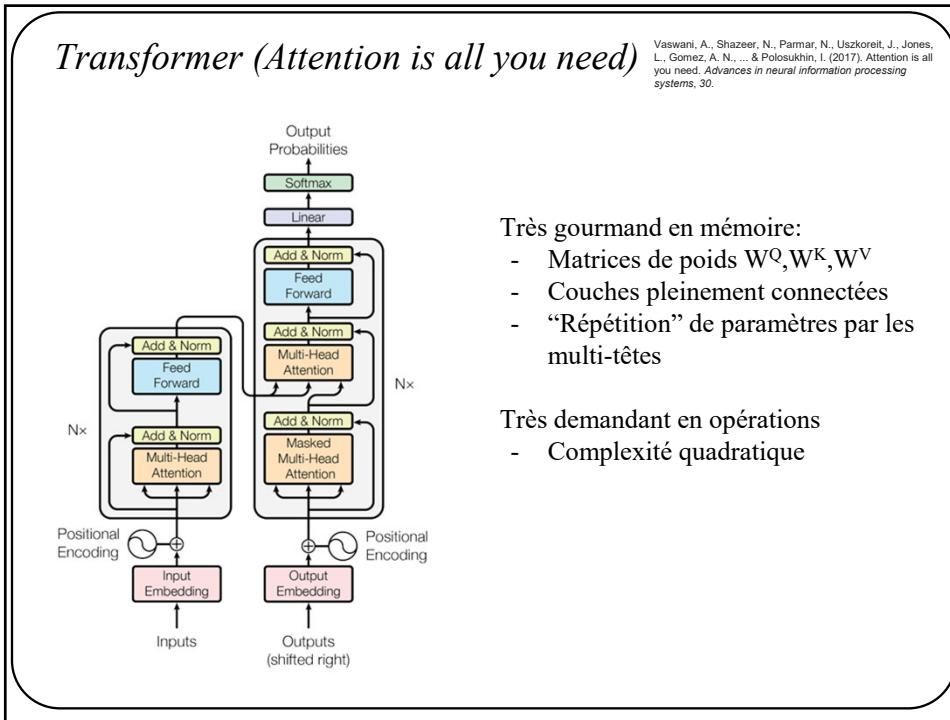
244



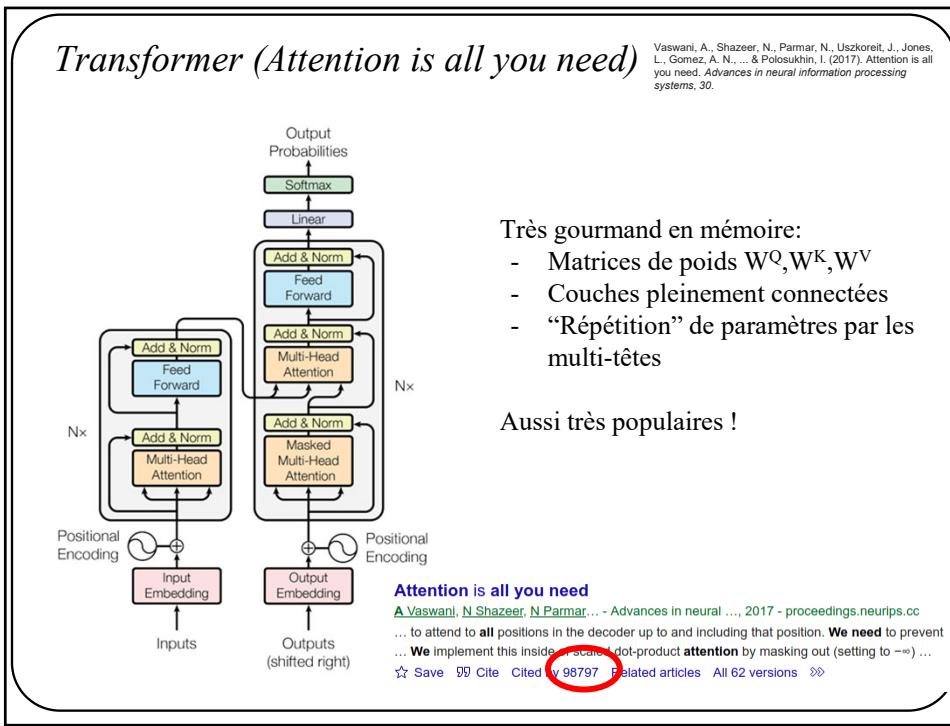
245



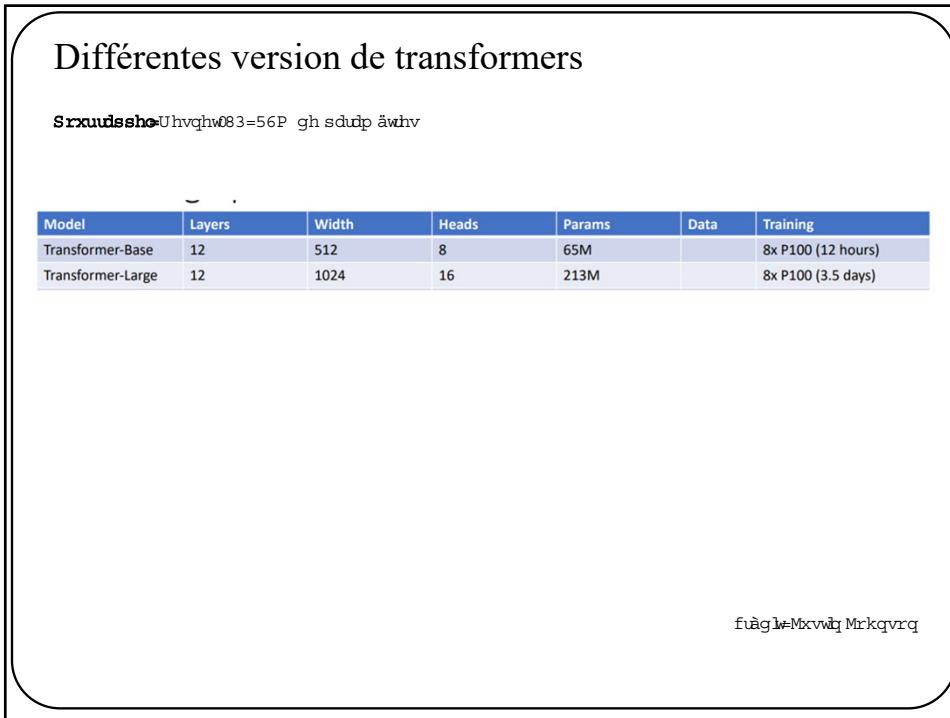
249



250



251



252

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	

fluid Mxvwq Mrkqvrc

253

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)

fluid Mxvwq Mrkqvrc

254

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	

fūāg l=Mxvñd Mrkqvrq

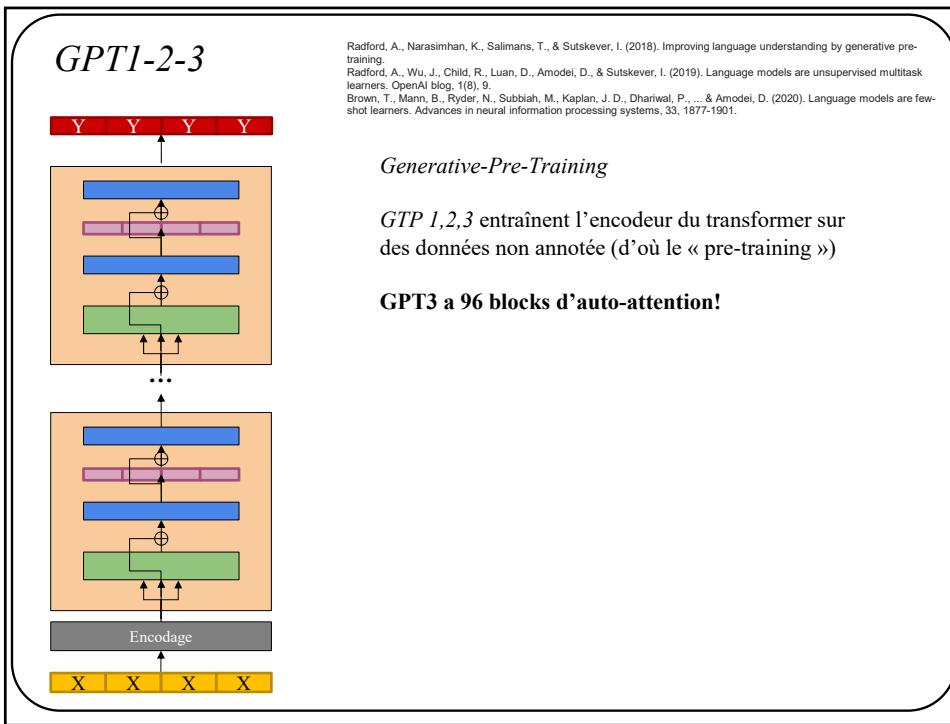
255

Transformers

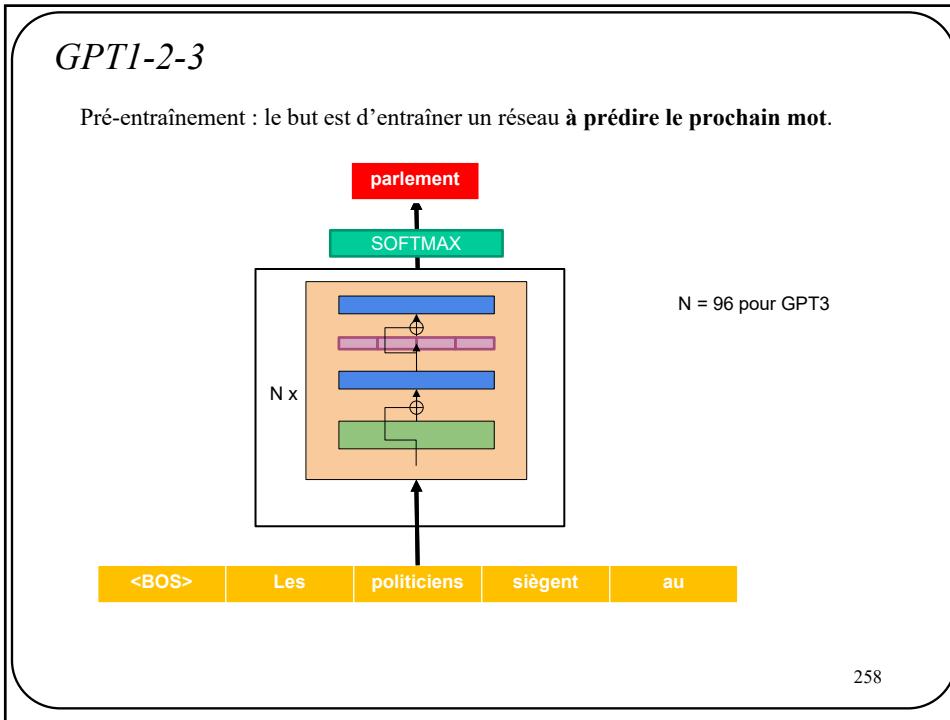
Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160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

fūāg l=Mxvñd Mrkqvrq

256



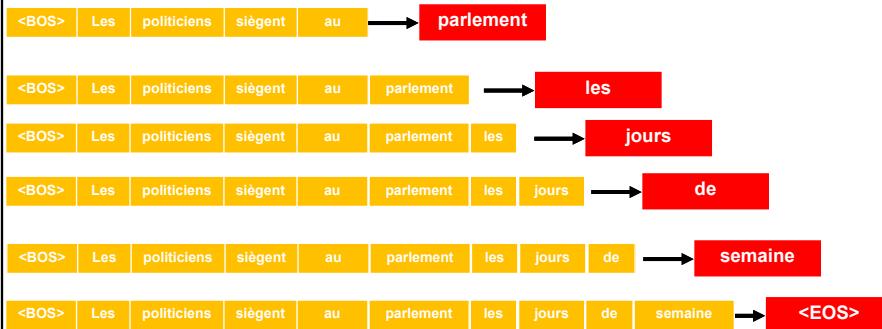
257



258

GPT1-2-3

Et on recommence:



259

259

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.

<https://openai.com/blog/better-language-models/#sample8>

260

Transformers

Les transformers sont *vraiment bons* pour générer du texte sémantiquement viable

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.

<https://app.inferkit.com/demo>

261

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.

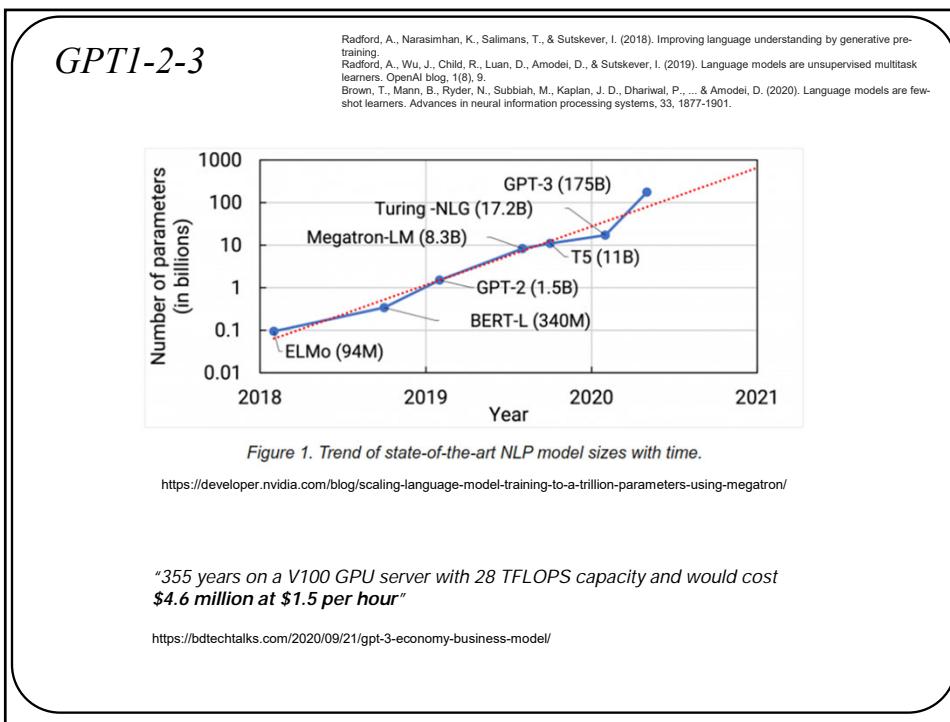
GPT-2
Parameters
117M
345M
762M
1542M
Layers
12
24
36
48
d_{model}
768
1024
1280
1600

Table 2. Architecture hyperparameters for the 4 model sizes.

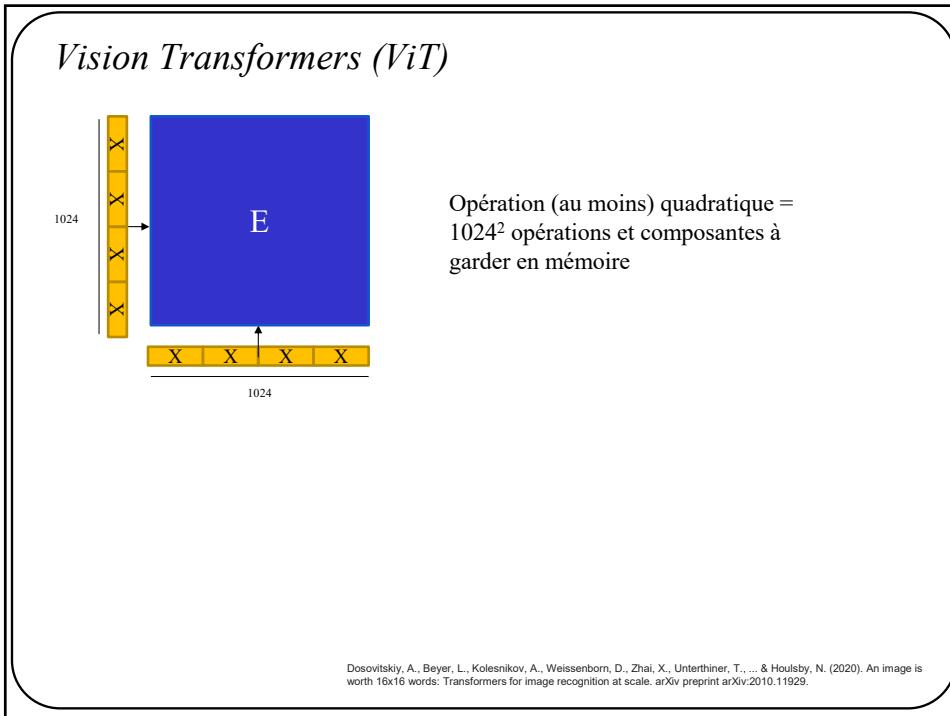
GPT-3
Model Name
n _{params}
n _{layers}
d_{model}
n _{heads}
d_{head}
Batch Size
Learning Rate
GPT-3 Small
125M
12
768
12
64
0.5M
6.0×10^{-4}
GPT-3 Medium
350M
24
1024
16
64
0.5M
3.0×10^{-4}
GPT-3 Large
760M
24
1536
16
96
0.5M
2.5×10^{-4}
GPT-3 XL
1.3B
24
2048
24
128
1M
2.0×10^{-4}
GPT-3.2.7B
2.7B
32
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.

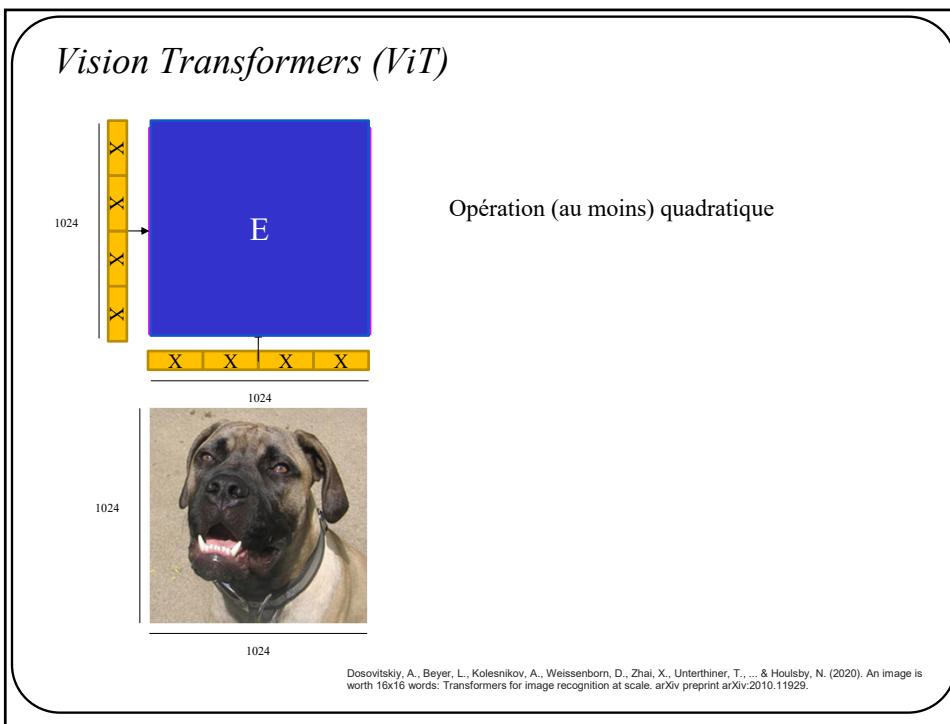
262



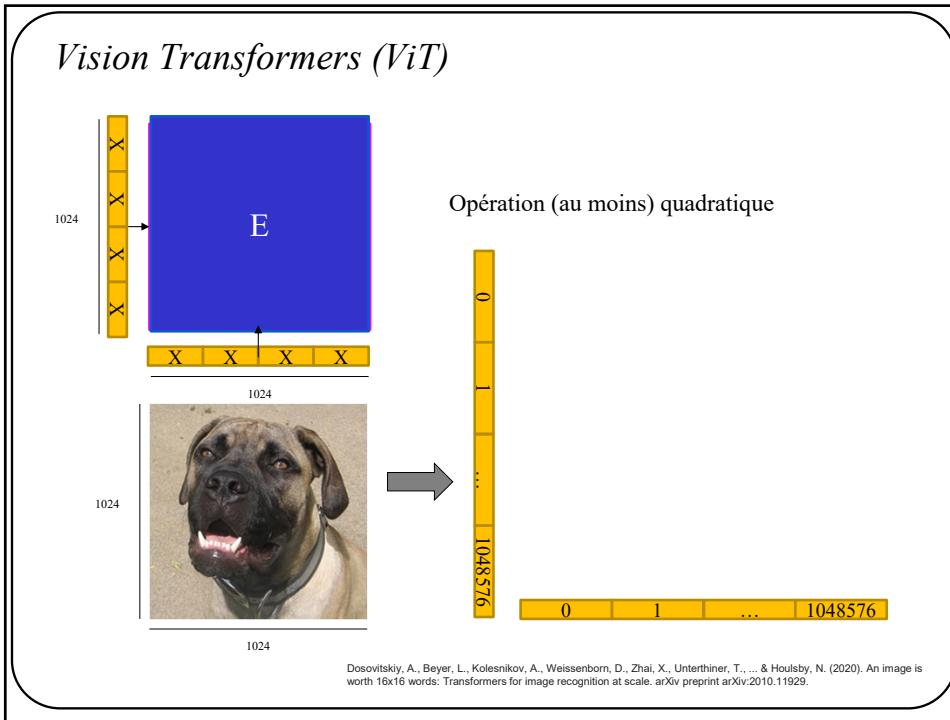
263



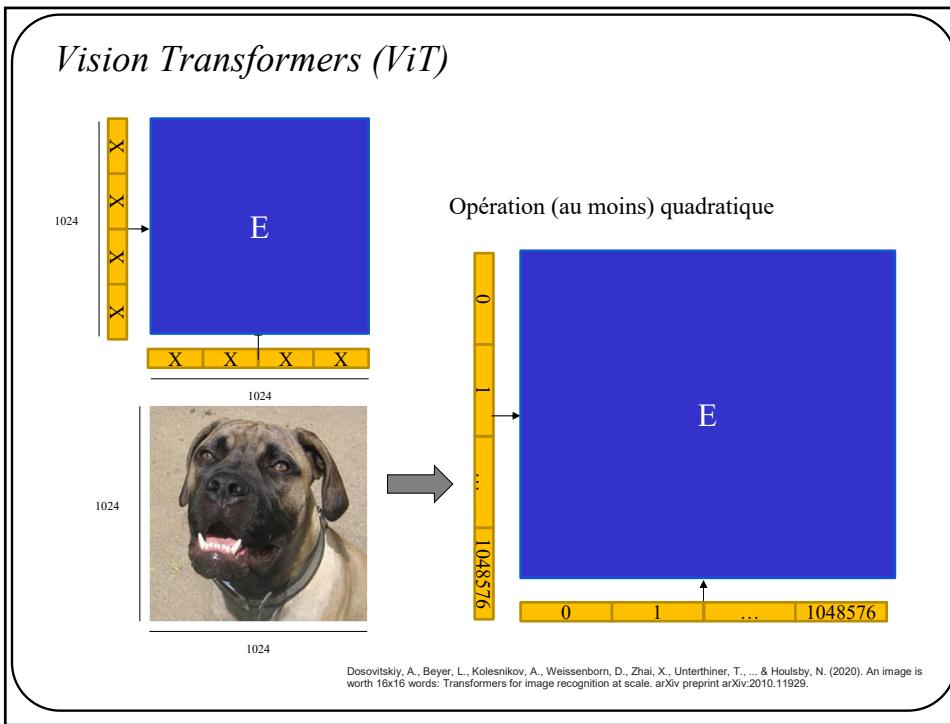
264



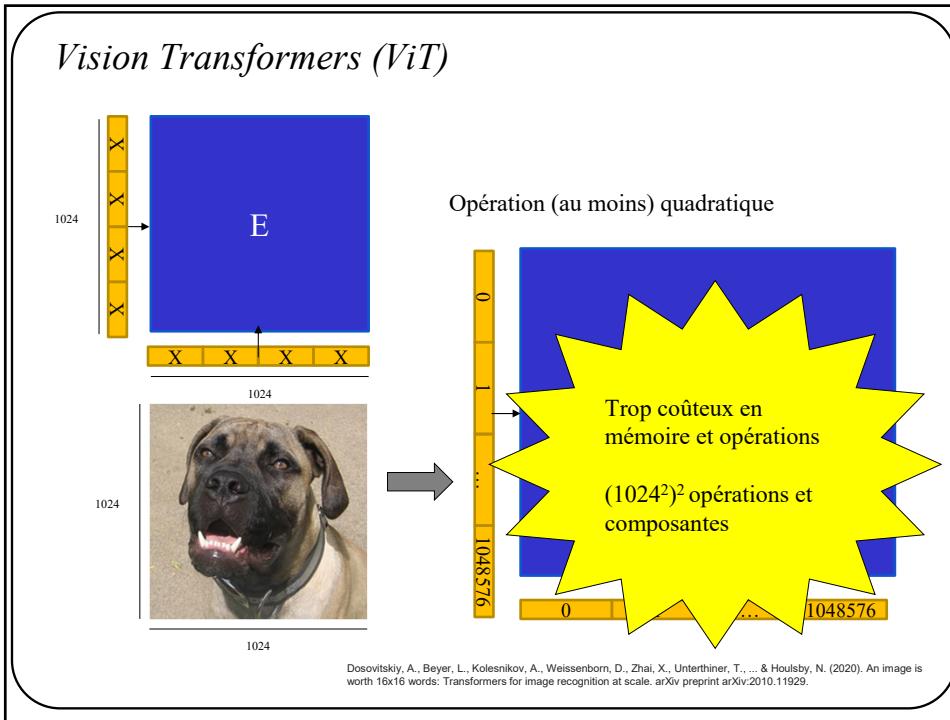
265



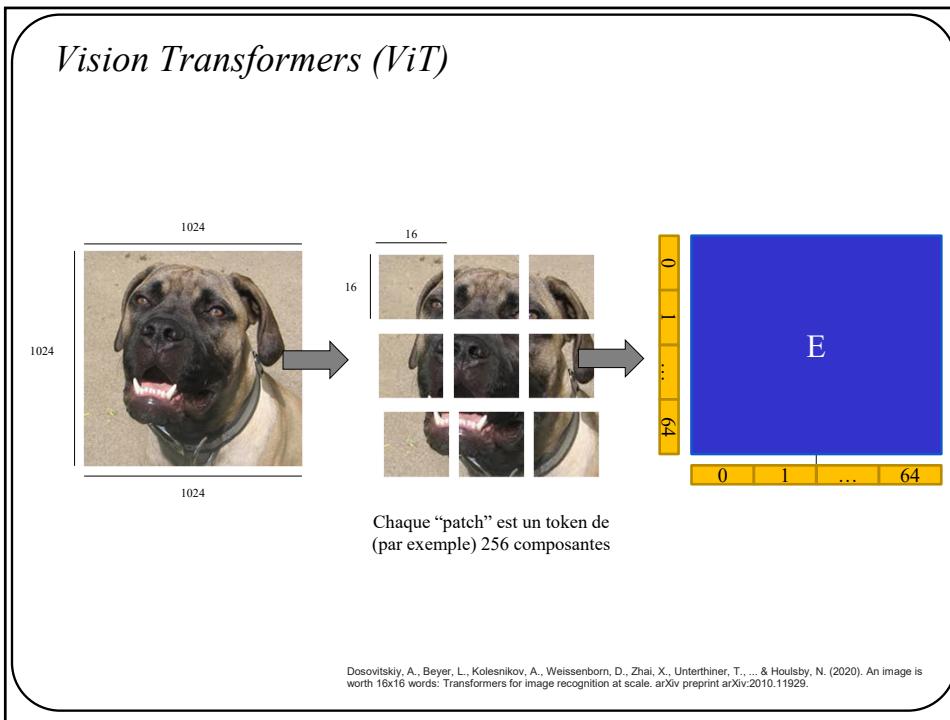
266



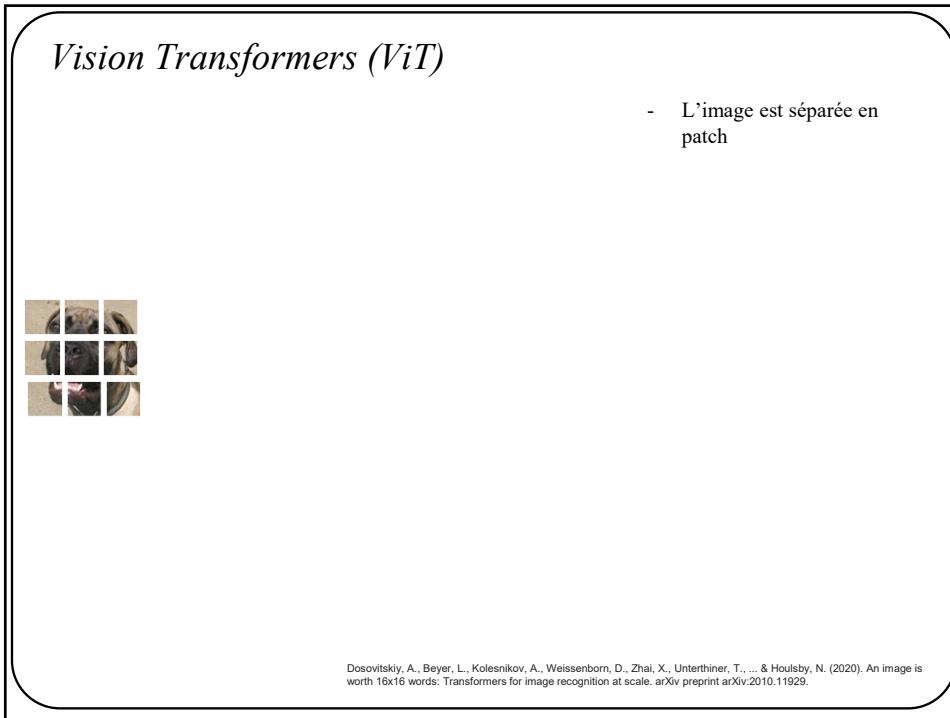
267



268



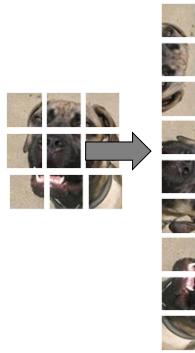
269



270

Vision Transformers (ViT)

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

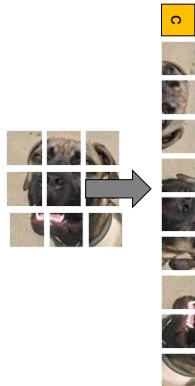


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

271

Vision Transformers (ViT)

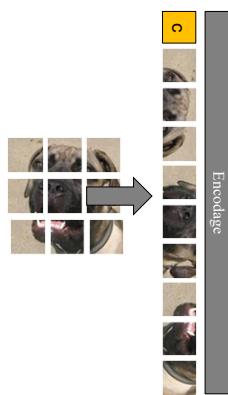
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté



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.

272

Vision Transformers (ViT)

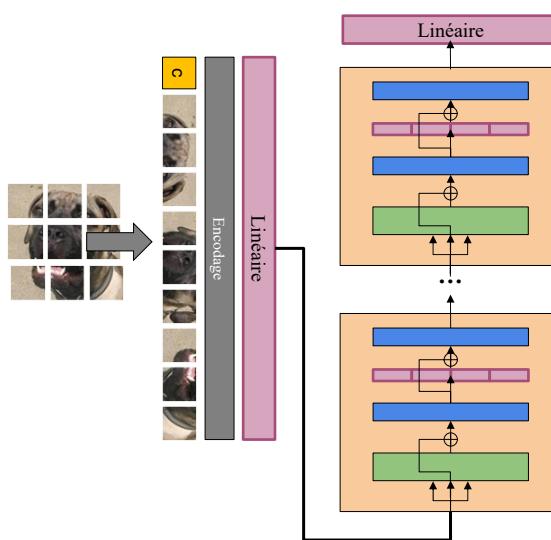


- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens

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.

273

Vision Transformers (ViT)

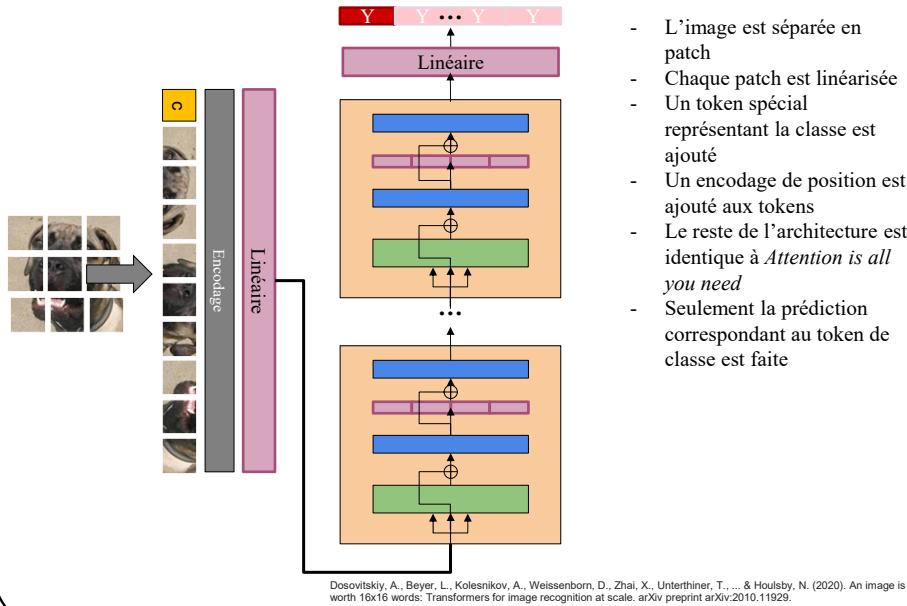


- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all you need*

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.

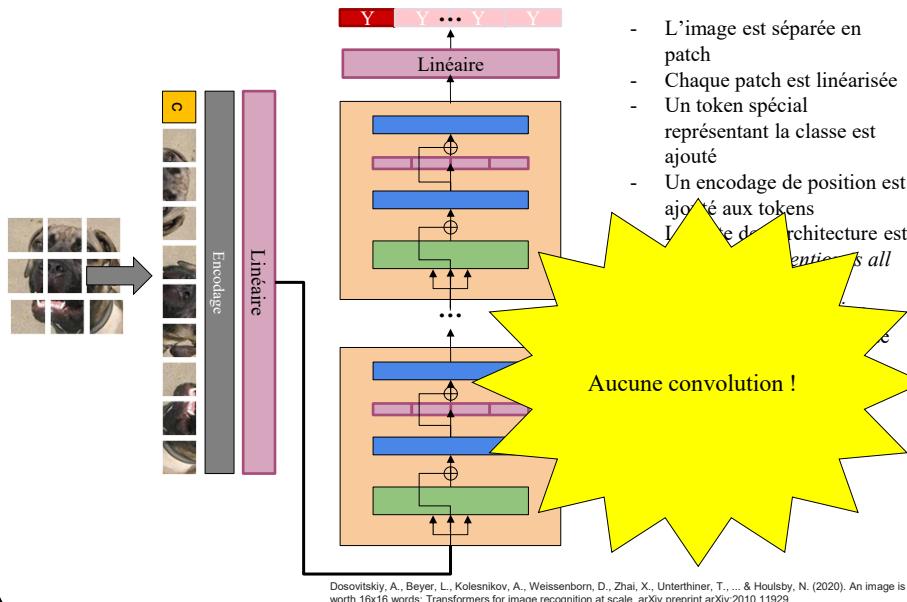
274

Vision Transformers (ViT)



275

Vision Transformers (ViT)



276

Vision Transformers (ViT)

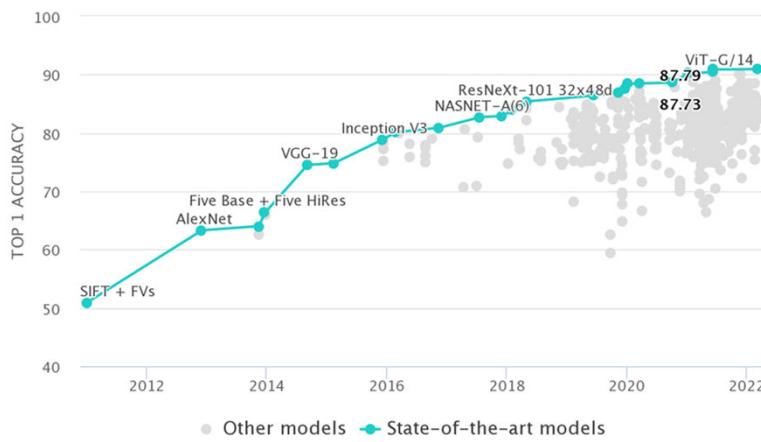
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4 / 88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 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).

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.

277

Vision Transformers (ViT)



Les vision transformers dominent la classification depuis leur arrivée

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

278

Vision Transformers (ViT)

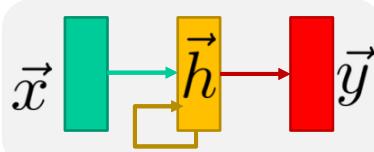
Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	Model soups (ViT-G/14)	90.94%		1843M	✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time	git	paper	2022	transformer pytorch
2	CoATNet-7	90.88%		2440M	✓	CoAtNet: Mixing Convolution and Attention for All Data Sizes	git	paper	2021	coatnet pytorch
3	ViT-G/14	90.45%		1843M	✓	Scaling Vision Transformers	git	paper	2021	transformer pytorch
4	CoATNet-6	90.45%		1470M	✓	CoAtNet: Mixing Convolution and Attention for All Data Sizes	git	paper	2021	coatnet pytorch
5	V-MeE-15B (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts	git	paper	2021	transformer
6	Meta Pseudo Labels (EfficientNet-L2)	90.2%	98.8%	480M	✓	Meta Pseudo Labels	git	paper	2021	efficientnet pytorch
7	SwinV2-G	90.17%			✓	Swin Transformer V2: Scaling Up Capacity and Resolution	git	paper	2021	transformer
8	Florence-CoSwin-H	90.05%	99.02%		✓	Florence: A New Foundation Model for Computer Vision	git	paper	2021	transformer
9	Meta Pseudo Labels (EfficientNet-B6-Wide)	90%	98.7%	390M	✓	Meta Pseudo Labels	git	paper	2021	efficientnet pytorch
10	NFNet-F4+	89.2%		527M	✓	High-Performance Large-Scale Image Recognition Without Normalization	git	paper	2021	nfnet pytorch

Les vision transformers dominent la classification depuis leur arrivée

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

279

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

tyntd-iafhatawaoihrdemot lytdws e ,tfti, astai f ogoh eosse rrranbyne 'nhthnee e
plia tklrgd t o idee ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomescerliund
Keushey. Thom here
sheulke, amerenith ol sivh I lalterthend Bleipile shuw fil on aseterlome
coaniogennc Phe lism thond hon at. MeDimorotion in ther thize."

↓ train more

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

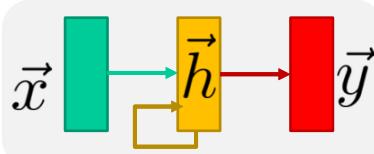
↓ train more

"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.

Modélisation de langage

280

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

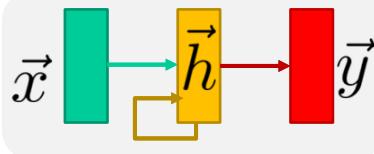


a group of people playing a game with nintendo wii controllers

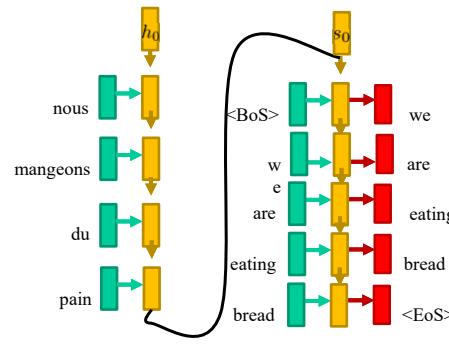
Description d'images

281

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



Traduction

282

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 utilisées pour faire une prédiction
- L'attention n'est pas limitée au texte, ou même aux séquences

283

Sommaire

- Un *Transformer* sont un modèle puissant pour les tâches liées au *langage naturel et aux images*
- Les *transformers* n'utilisent *que* l'attention (pas un modèle récurrent)
- Les *transformers* sont demandant en ressources

284