Réseaux de neurones

IFT 780

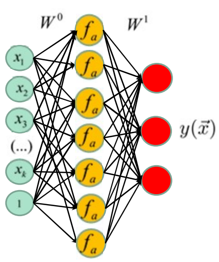
Réseaux récurrents

Par

Pierre-Marc Jodoin, Antoine Théberge

1

Réseau de neurones de base (régression)



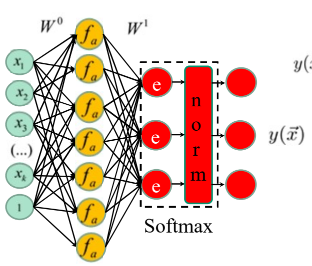
$$y(\vec{x}) = W^1 f_a(W^0\vec{x})$$

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

f_a : fonction d'activation

2

Réseau de neurones de base (classification)

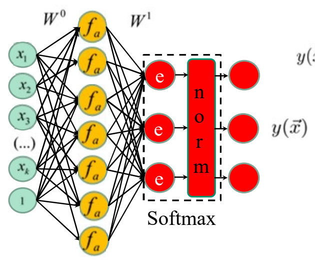


$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0\vec{x}))$$

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

3

Réseau de neurones de base (classification)



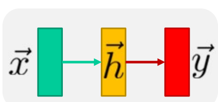
$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0\vec{x}))$$

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

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

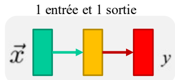


RN de base

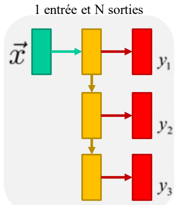
5

Différentes configurations pour différentes applications

1 entrée et 1 sortie

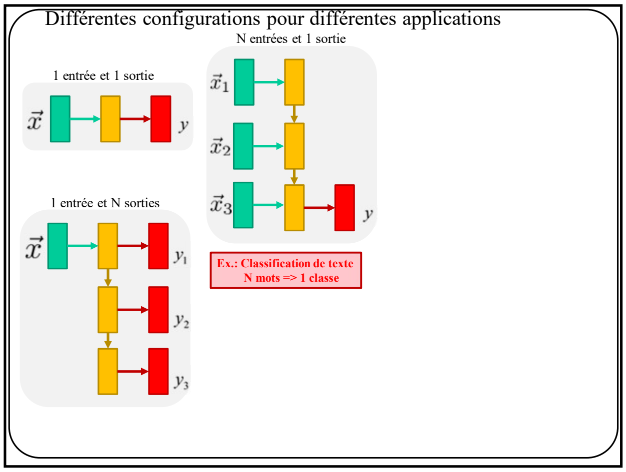


1 entrée et N sorties

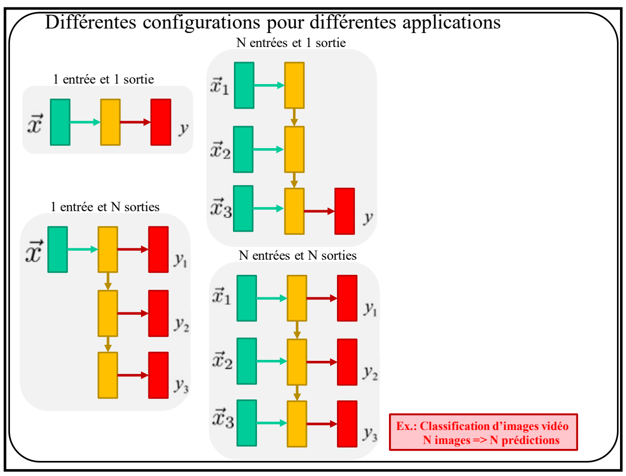


Ex.: description d’une image
1 image => N mots

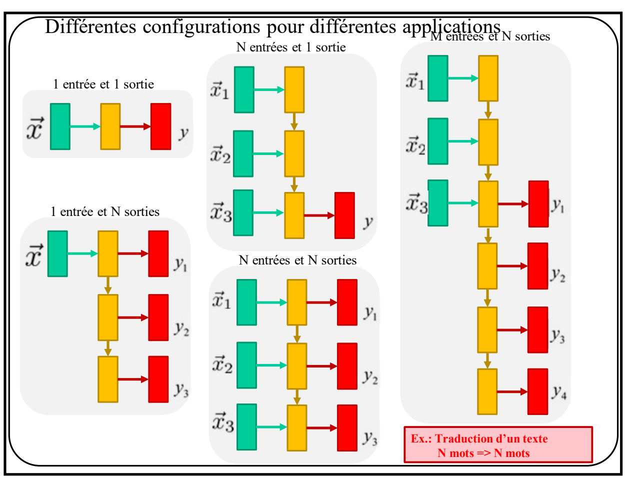
6


Différentes configurations pour différentes applications
1 entrée et 1 sortie
N entrées et 1 sortie
1 entrée et N sorties
Ex.: Classification de texte
N mots => 1 classe

7

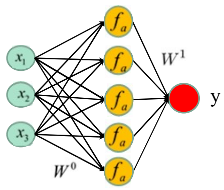

Différentes configurations pour différentes applications
1 entrée et 1 sortie
N entrées et 1 sortie
1 entrée et N sorties
N entrées et N sorties
Ex.: Classification d'images vidéo
N images => N prédictions

8


Différentes configurations pour différentes applications
1 entrée et 1 sortie
N entrées et 1 sortie
M entrées et N sorties
1 entrée et N sorties
N entrées et N sorties
Ex.: Traduction d'un texte
N mots => N mots

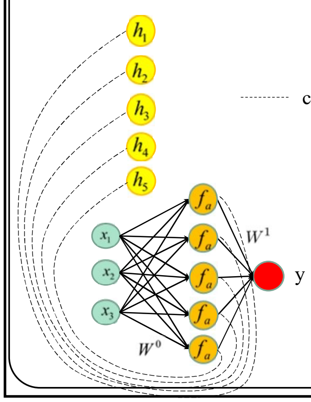
9

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



10

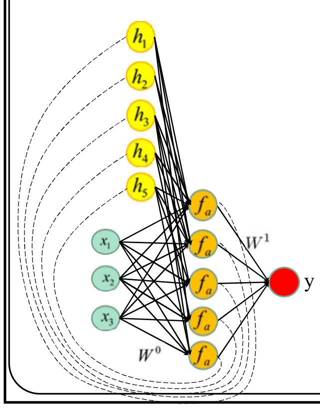
Réseau récurrent : la sortie des neurones est réinjectée dans leur entrée



copie

11

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

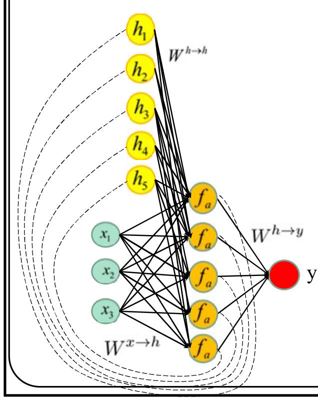


copie

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

12

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



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

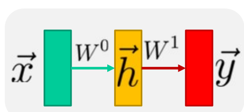
$$\downarrow$$

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

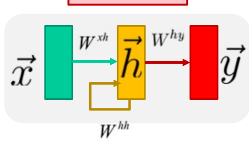
$$y(\vec{x}) = W^{h\to y}\vec{h}$$

13

Illustration simplifiée



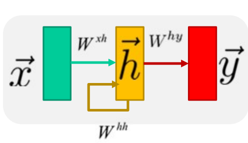
RN de base



RN récurrent

14

Dans le cas général avec K sorties (régression)



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

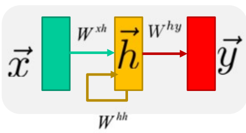
$$\downarrow$$

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

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

15

Dans le cas général avec K sorties (classification)



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

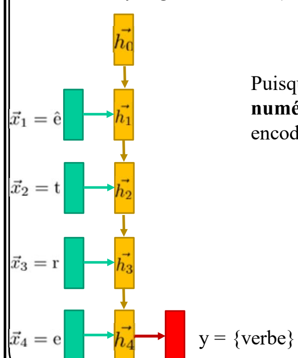
$$\downarrow$$

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

16

Exemple pour N entrées et 1 sortie:

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



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

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

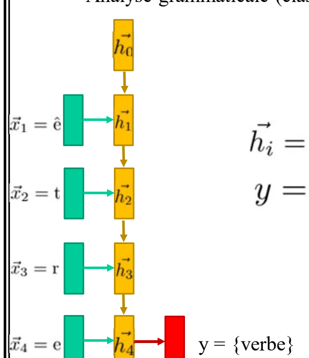
....

$$\left.\begin{array}{l} '\textit{verbe}' = [1,0,0\ldots0] \\ '\textit{nom}' = [0,1,0,\ldots0] \\ '\textit{adjectif}' = [0,0,1,\ldots0] \end{array}\right\} \in R^{M}$$

17

Exemple pour N entrées et 1 sortie:

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

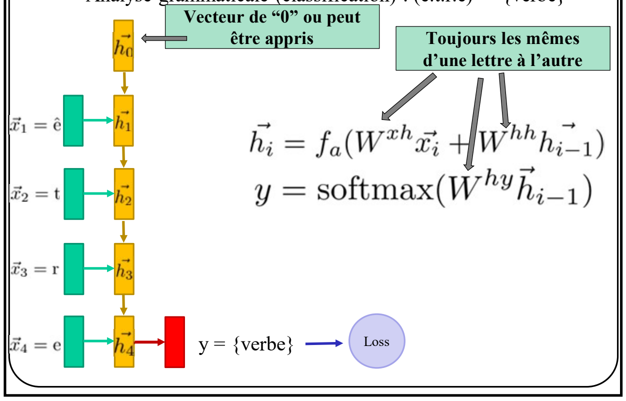


$$\vec{h_i} = f_a(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$
$$y = \text{softmax}(W^{hy}\vec{h}_{i-1})$$

18

Exemple pour N entrées et 1 sortie:

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

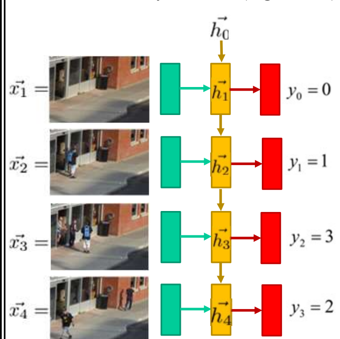


$$\vec{h}_i = f_a(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$
$$y = \mathrm{softmax}(W^{hy}\vec{h}_{i-1})$$

19

Même idée pour N entrées et N sorties:

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

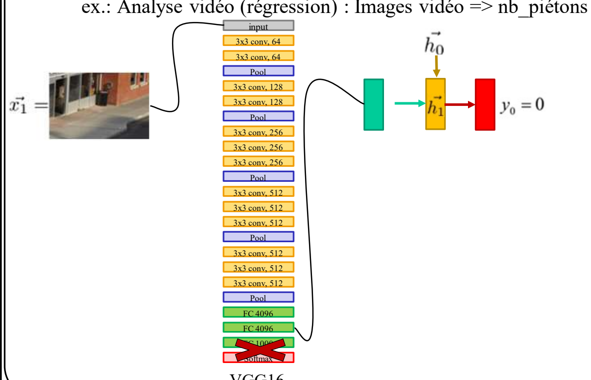


$$\vec{h}_i = f_a(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1})$$
$$y_i = W^{hy}\vec{h}_i$$

20

Même idée pour N entrées et N sorties:

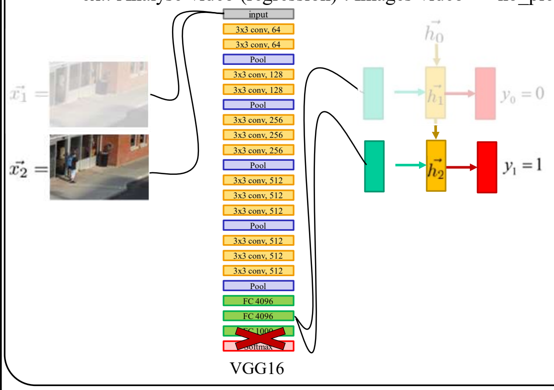
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



21

Même idée pour N entrées et N sorties:

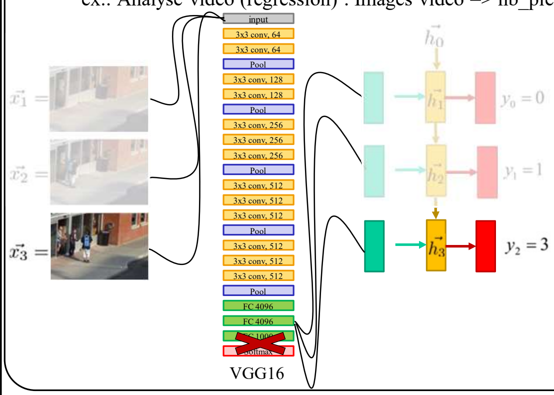
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



22

Même idée pour N entrées et N sorties:

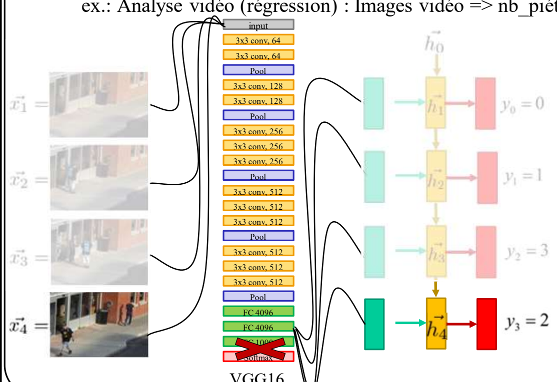
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



23

Même idée pour N entrées et N sorties:

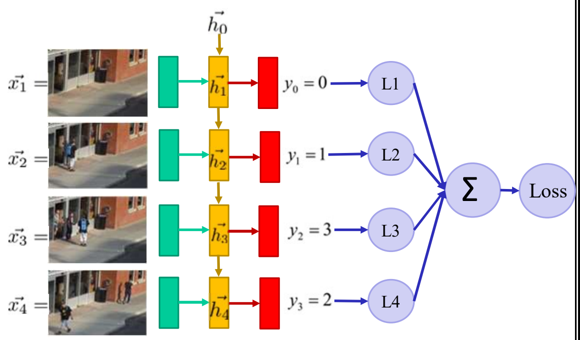
ex.: Analyse vidéo (régression) : Images vidéo => nb_piétons



24

Même idée pour N entrées et N sorties:

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



25

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

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

Représentation « one hot » jouet:

‘a’ = [1, 0, 0, 0]
‘e’ = [0, 1, 0, 0]
‘m’ = [0, 0, 1, 0]
‘s’ = [0, 0, 0, 1]

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

26

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

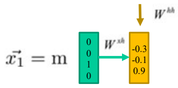
Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

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

27

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

Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

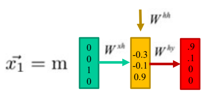


$$\vec{h_i} = \tanh(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$

28

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

Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

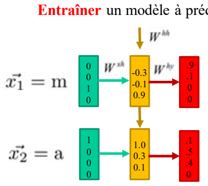


$$\vec{h_i} = \tanh(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$
$$\vec{y}(\vec{x_i}) = \mathrm{softmax}(W^{hy}\vec{h_i})$$

29

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

Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

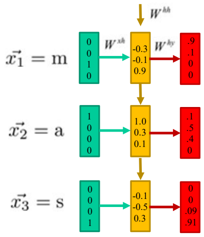


$$\vec{h_i} = \tanh(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$
$$\vec{y}(\vec{x_i}) = \mathrm{softmax}(W^{hy}\vec{h_i})$$

30

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

Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

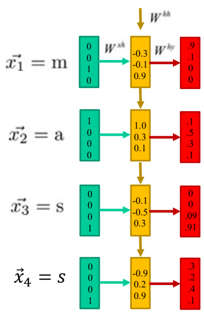


$$\vec{h_i} = \tanh(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$
$$\vec{y}(\vec{x_i}) = \text{softmax}(W^{hy}\vec{h_i})$$

31

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

Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».

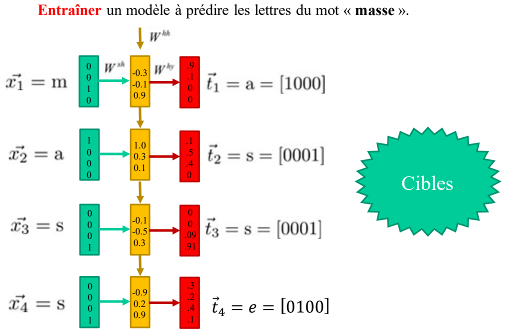


$$\vec{h_i} = \tanh(W^{xh}\vec{x_i} + W^{hh}\vec{h_{i-1}})$$
$$\vec{y}(\vec{x_i}) = \text{softmax}(W^{hy}\vec{h_i})$$

32

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

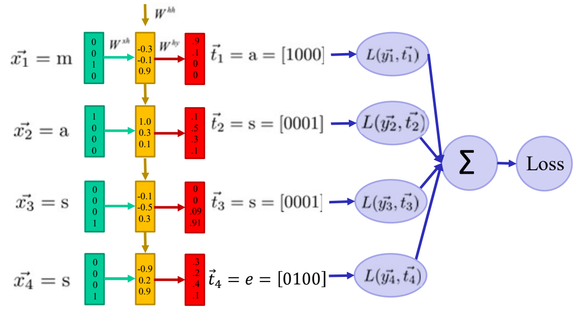
Alphabet :[**a,e,m,s**]
Entraîner un modèle à prédire les lettres du mot « **masse** ».



33

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

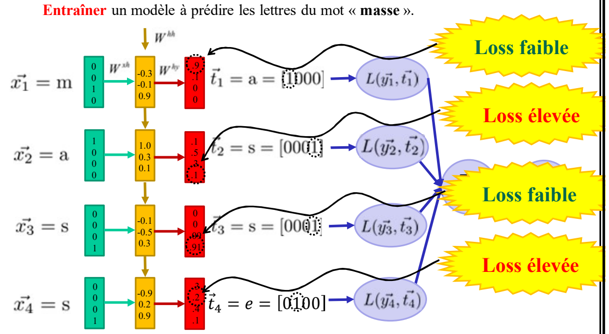
Alphabet :**[a,e,m,s]**
Entraîner un modèle à prédire les lettres du mot « **masse** ».



34

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

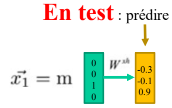
Alphabet :**[a,e,m,s]**
Entraîner un modèle à prédire les lettres du mot « **masse** ».



35

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

Alphabet :**[a,e,m,s]**
En test : prédire les lettres les unes après les autres



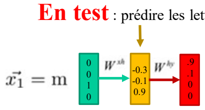
Étape 1 : Calcul de la couche cachée

36

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

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

En test : prédire les lettres les unes après les autres



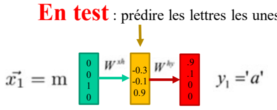
Étape 2 : Calcul de la sortie (softmax)

37

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

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

En test : prédire les lettres les unes après les autres



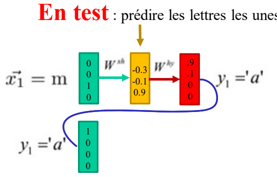
Étape 3 : Sélectionner le caractère le plus probable

38

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

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

En test : prédire les lettres les unes après les autres



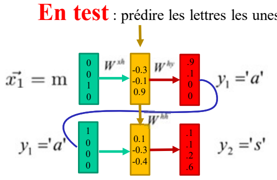
Étape 4 : Injecter le caractère prédit au début du réseau

39

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

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

En test : prédire les lettres les unes après les autres



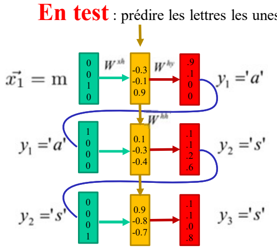
Et on recommence!

40

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

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

En test : prédire les lettres les unes après les autres

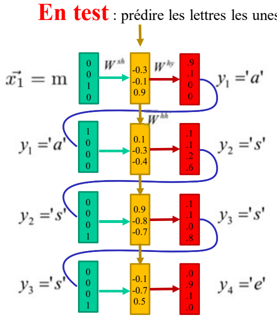


41

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

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

En test : prédire les lettres les unes après les autres



42

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

Code python: **“mini-char-RNN” de A. Karpathy**
https://gist.github.com/karpathy/d4dee566867f8291f086
Un RNN en 112 lignes !

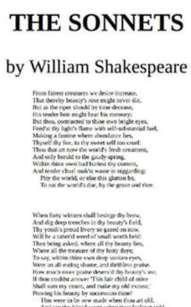
$$\left.\begin{aligned} 'a' &= [1,0,0,\dots,0] \\ 'b' &= [0,1,0,\dots,0] \\ 'c' &= [0,0,1,\dots,0] \end{aligned}\right\} \in R^{256}$$

...

43

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

Code python: **“mini-char-RNN” de A. Karpathy**
https://gist.github.com/karpathy/d4dee566867f8291f086



THE SONNETS

by William Shakespeare

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

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

44

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

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

↓ train more

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

↓ train more

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

↓ train more

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

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 srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

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

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

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

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

VIOLA:
I'll drink it.
```

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

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

Crédit: A. Karpathy, CS231

46

Entraînement sur le code source de Linux en C++

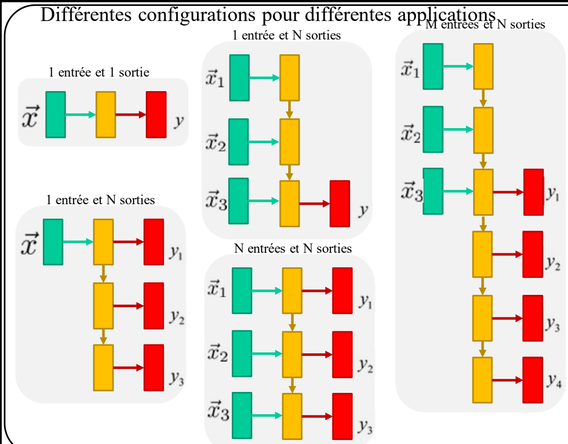
```c
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* 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 ");
}
```

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

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

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

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

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

Crédit: A. Karpathy, CS231

47

Différentes configurations pour différentes applications

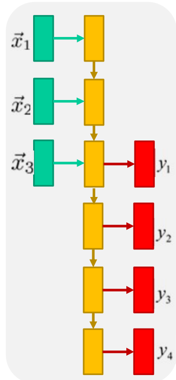


48

Traduction Français-Anglais

N lettres => M lettres

M entrées et N sorties



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>]**

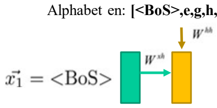
Pas le même nombre d'entrées que de sorties !
(BoS : Begining of Sentence, EoS:End of Sentence).

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>]**

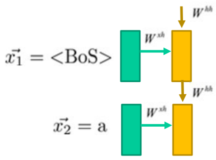


51

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>]**

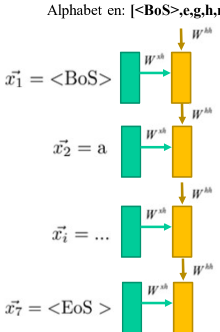


52

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>]**

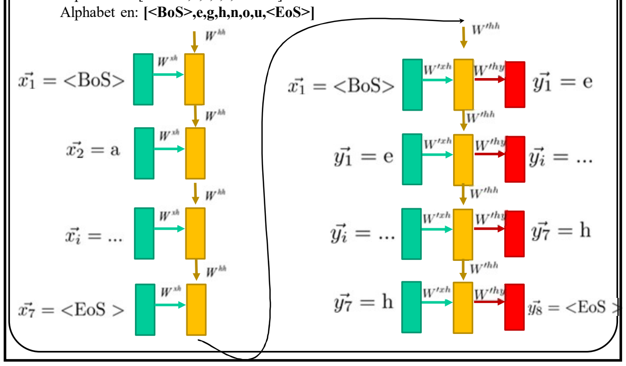


53

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>]**

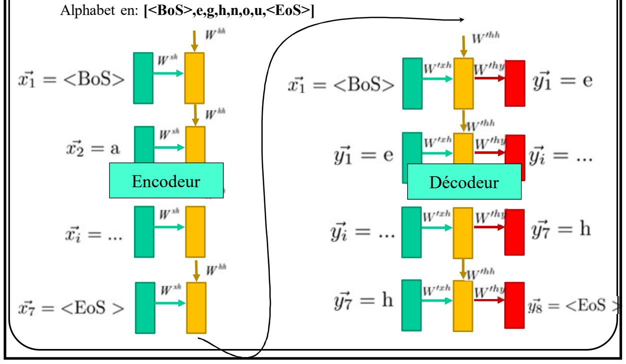


54

Autre exemple: **traduction**

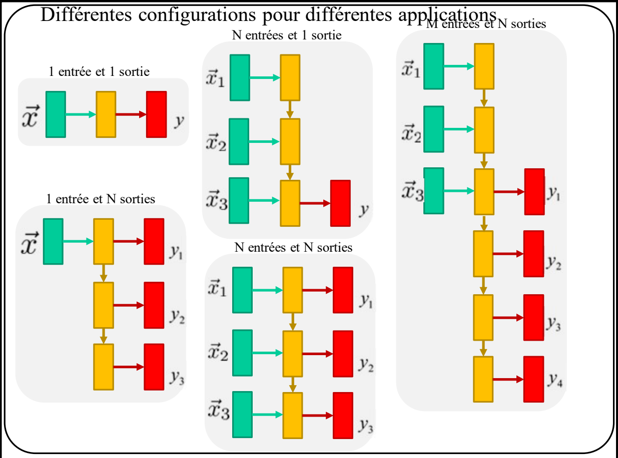
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Traduire 'assez' -> 'enough'
Alphabet fr :[**<BoS>,a,e,s,z,<EoS>**]
Alphabet en: **[<BoS>,e,g,h,n,o,u,<EoS>]**



55

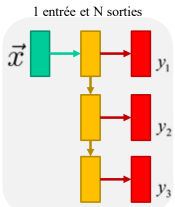
Différentes configurations pour différentes applications



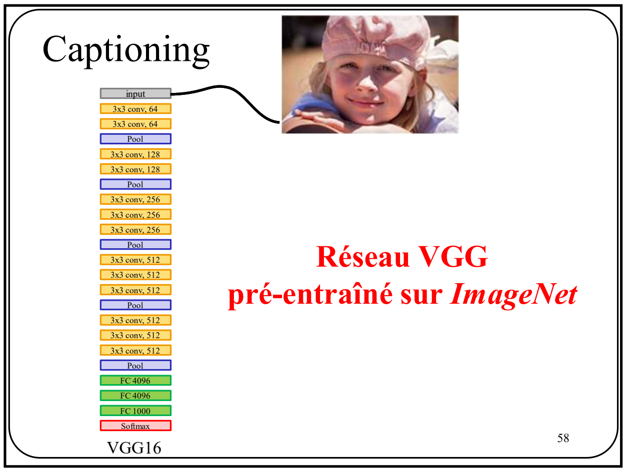
56

Différentes configurations pour différentes applications

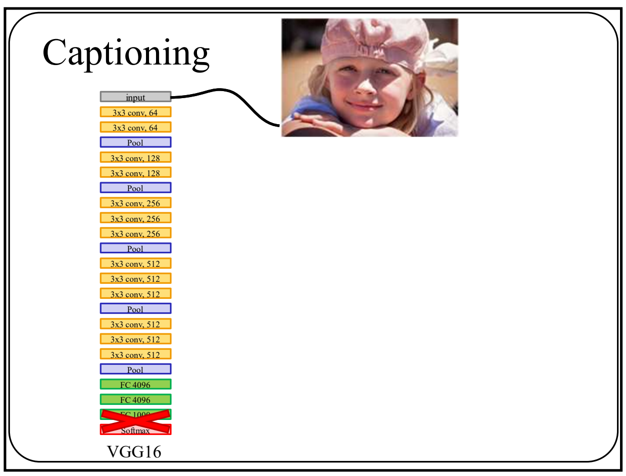
Description du contenu d'une image (« captioning »)



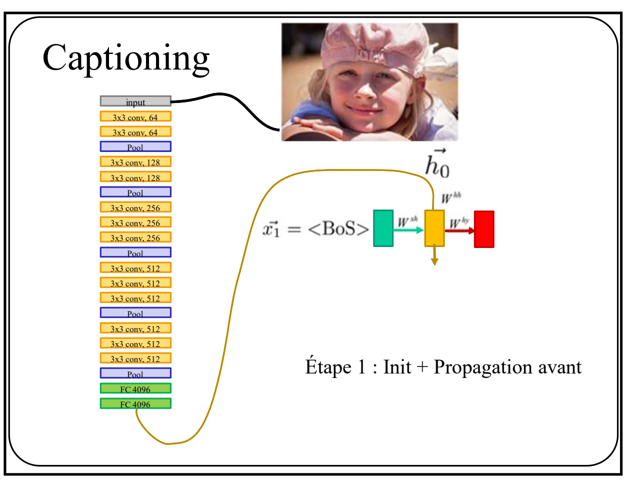
57


Captioning
input
3x3 conv, 64
3x3 conv, 64
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
FC 4096
FC 4096
FC 1000
Softmax
VGG16
Réseau VGG
pré-entraîné sur *ImageNet*
58

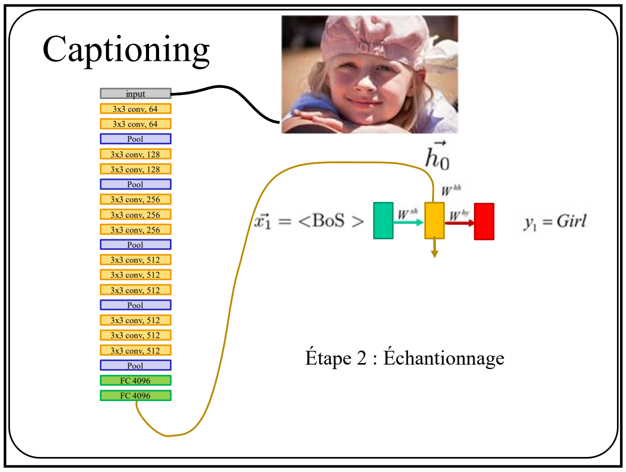
58


Captioning
input
3x3 conv, 64
3x3 conv, 64
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
FC 4096
FC 4096
~~FC 1000~~
~~Softmax~~
VGG16

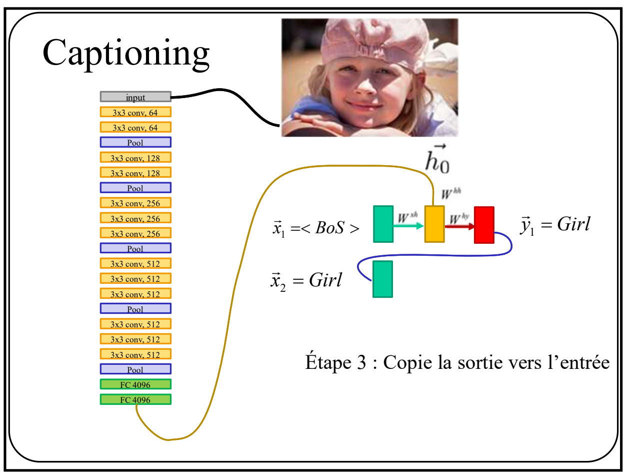
59


Captioning
input
3x3 conv, 64
3x3 conv, 64
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
FC 4096
FC 4096
$\vec{h}_0$
W^{hh}
$\vec{x}_1$ = <BoS>
W^{xh}
W^{hy}
Étape 1 : Init + Propagation avant

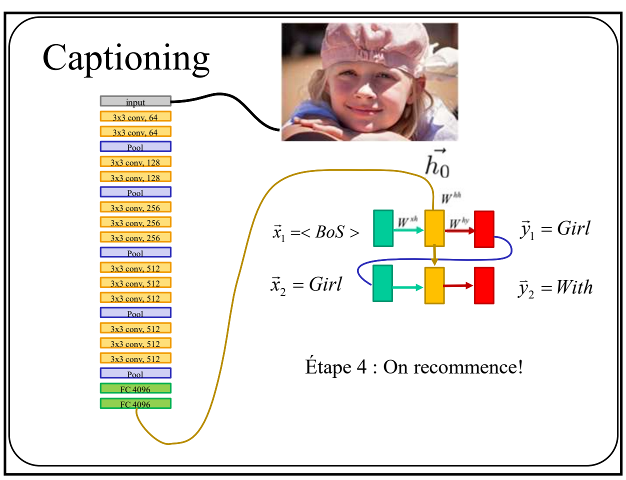
60


Captioning
$\vec{h}_0$
W^{hh}
$\vec{x}_1$ = <BoS >
W^{xh}
W^{hy}
$y_1 = Girl$
Étape 2 : Échantionnage

61

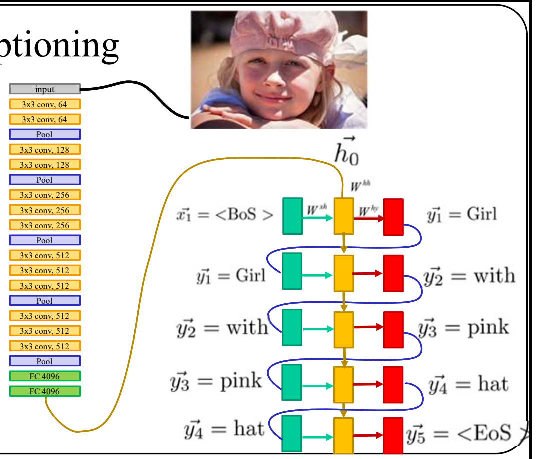

Captioning
$\vec{h}_0$
W^{hh}
$\vec{x}_1$ =< BoS >
W^{xh}
W^{hy}
$\vec{y}_1 = Girl$
$\vec{x}_2 = Girl$
Étape 3 : Copie la sortie vers l'entrée

62


Captioning
$\vec{h}_0$
W^{hh}
$\vec{x}_1$ =< BoS >
W^{xh}
W^{hy}
$\vec{y}_1 = Girl$
$\vec{x}_2 = Girl$
$\vec{y}_2 = With$
Étape 4 : On recommence!

63

Captioning



64

Exemples de résultats

https://github.com/karpathy/neuraltalk2


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


a man riding a wave on top of a surfboard

a street sign on a pole in front of a building


a group of people playing a game with nintendo wii controllers


a couple of zebra standing on top of a dirt field

65

65

Exemples d'erreurs

https://github.com/karpathy/neuraltalk2


a man is throwing a frisbee in a park


a man riding a skateboard down a street


a laptop computer sitting on top of a wooden desk


A woman is holding a cat in her hand


A woman standing on a beach holding a surfboard

66

66

NeuralTalk and Walk https://vimeo.com/146492001

67

67

Analyse de texte

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

Pour des **caractères**…

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

68

68

Analyse de texte

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

Pour des **mots**…

$$\ldots$$
$$\left.\begin{array}{l} '\mathit{grand}' = [\ldots,1,0,0,\ldots,0] \\ '\mathit{grandement}' = [\ldots,0,1,0,\ldots,0] \\ '\mathit{grandeur}' = [\ldots,0,0,1,\ldots,0] \end{array}\right\} \in R^{10,000}$$
$$\ldots$$

69

69

Prédiction sur des lettres vs. mots

$'a' = [1,0,0,...,0]$
$'b' = [0,1,0,...,0]$
$'c' = [0,0,1,...,0]$
...

$\in R^{256}$ Prédiction sur des lettres

...
$'grand' = [...,1,0,0,...,0]$
$'grandement' = [...,0,1,0,...,0]$
$'grandeur' = [...,0,0,1,...,0]$
...

$\in R^{10,000}$ Prédiction sur des mots

70

Prédiction sur des lettres vs. mots

$'a' = [1,0,0,...,0]$
$'b' = [0,1,0$
$'c' = [0,0,1$
...

...
$'grand' =$
$'grandem$
$'grandeu$
...

lettres

mots

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

71

On peut aussi utiliser des fractions de mots

$$\left.\begin{array}{r} \dots \\ 'e' = [0,0,...,1,..,0] \\ \dots \\ 'grand' = [0,0,...,1,..,0] \\ \dots \\ 'ment' = [0,0,...,1,..,0] \\ \dots \end{array}\right\} \in \mathbb{R}^m$$

'grand'
'grand'+'e'
'grand'+'e'+'ment'

72

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 »*

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

Dictionnaire de Jetons

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

1 ligne = code pour 1 mot

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 »

Dictionnaire Jeton (matrice d’encodage)

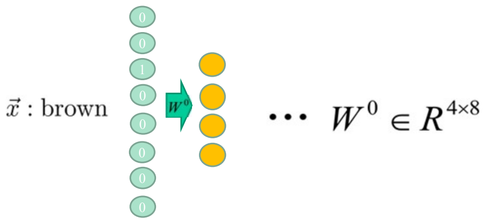
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 3 & 4 & 5 \\ -1 & -3 & -2 & 2 \\ 11 & 6 & 4 & -3 \\ -4 & 8 & -4 & 4 \\ 24 & -6 & 42 & 17 \\ 91 & 13 & 14 & -5 \\ 0 & 36 & 4 & 56 \\ -1 & 0 & 1 & 35 \end{pmatrix} = \begin{pmatrix} 11 & 6 & 4 & -3 \end{pmatrix}$$

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

$$jeton_{\vec{x}} = W^0 \; \vec{x}$$

78

Word2Vec s'appuie sur 2 idées fondamentales

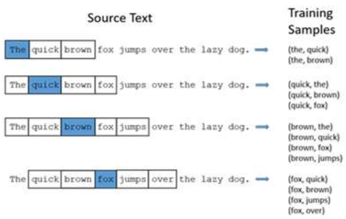
Idée 1: **Dictionnaire = matrice d'encodage**

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

79

Word2Vec s'appuie sur 2 idées fondamentales

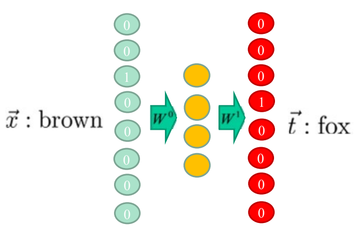
Idée 2: **2 mots proches dans un texte = 2 mots proches sémantiquement**



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

80

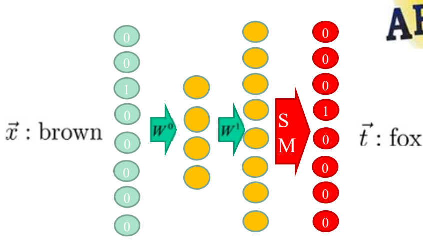
Word2Vec [Mikolov et al. '13]



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

81

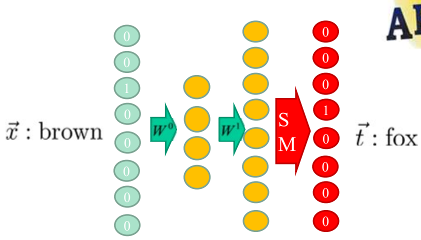
Word2Vec [Mikolov et al. '13]



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

82

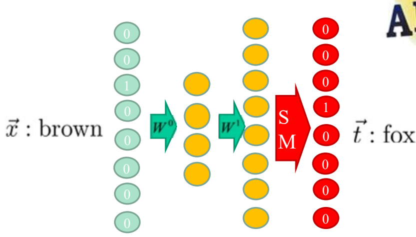
Word2Vec [Mikolov et al. ’13]



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

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	0.79		maddyness	0.77
	env windows	0.79		tweet	0.77

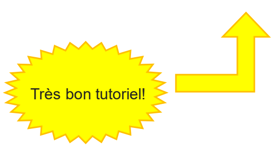
Ahmia, Oussama & Béchet, Nicolas & Marteau, Pierre-Francois. *Two Multilingual Corpora Extracted from the Tenders Electronic Daily for Machine Learning and Machine Translation Applications*.in LREC 2018

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

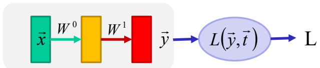
87

Comment entraîner un RNN?

88

Histoire de gradients

RN de classification avec entropie croisée

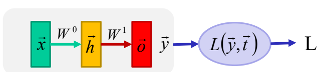


$$\vec{y}(\vec{x}) = S_M\left(W^1 \tanh\left(W^0 \vec{x}\right)\right)$$
$$L = L_{EC}\left(\vec{y},\vec{t}\right)$$

89

Histoire de gradients

Simple RN de classification avec entropie croisée



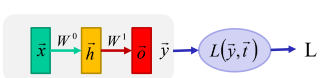
$$\vec{h} = \tanh\left(W^0 \vec{x}\right)$$
$$\vec{o} = W^1 \vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

Propagation avant

90

Histoire de gradients

Simple RN de classification avec entropie croisée



$$\vec{h} = \tanh\left(W^0 \vec{x}\right)$$
$$\vec{o} = W^1 \vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

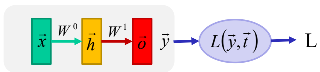
Pour entraîner le réseau il faut calculer

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

91

Histoire de gradients

Simple RN de classification avec entropie croisée



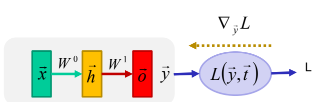
$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

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



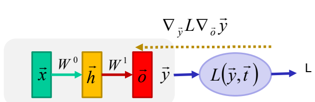
$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

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

Rétro-propagation

93

Histoire de gradients



$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

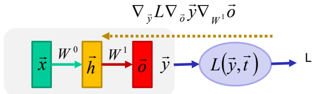
$$\nabla_{\vec{o}} \vec{y} = I\vec{y}^T - \vec{y}^T\vec{y}$$
$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-propagation

94

Histoire de gradients

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



$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

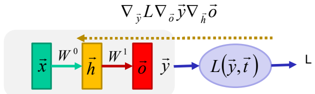
$$\nabla_{W^1}\vec{o} = \vec{h}$$
$$\nabla_{\vec{o}}\vec{y} = \mathrm{I}\vec{y}^T - \vec{y}^T\vec{y}$$
$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

95

Histoire de gradients

$$\nabla_{\vec{y}} L \nabla_{\vec{o}} \vec{y} \nabla_{\vec{h}} \vec{o}$$



$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

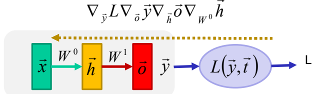
$$\nabla_{\vec{h}}\vec{o} = W^1$$
$$\nabla_{W^1}\vec{o} = \vec{h}$$
$$\nabla_{\vec{o}}\vec{y} = \mathrm{I}\vec{y}^T - \vec{y}^T\vec{y}$$
$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

96

Histoire de gradients

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



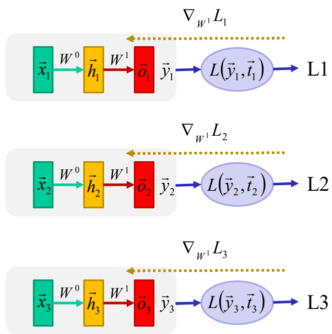
$$\vec{h} = \tanh\left(W^0\vec{x}\right)$$
$$\vec{o} = W^1\vec{h}$$
$$\vec{y} = S_M\left(\vec{o}\right)$$
$$L = L_{CE}\left(\vec{y},\vec{t}\right)$$

$$\nabla_{W^o}\vec{h} = 1 - \tanh^2\left(W^1\vec{x}\right)\vec{x}$$
$$\nabla_{\vec{h}}\vec{o} = W^1$$
$$\nabla_{W^1}\vec{o} = \vec{h}$$
$$\nabla_{\vec{o}}\vec{y} = \mathrm{I}\vec{y}^T - \vec{y}^T\vec{y}$$
$$\nabla_{\vec{y}} L = -\frac{\vec{t}}{\vec{y}}$$

Rétro-
propagation

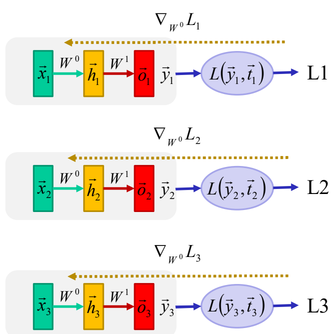
97

Ex.: 3 données, 3 rétro-propagations



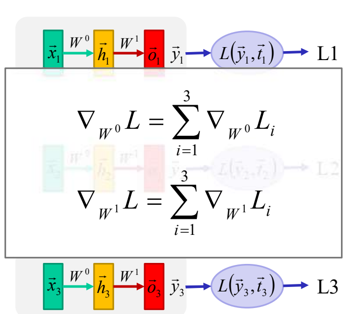
98

Ex.: 3 données, 3 rétro-propagations



99

3 rétro-propagations

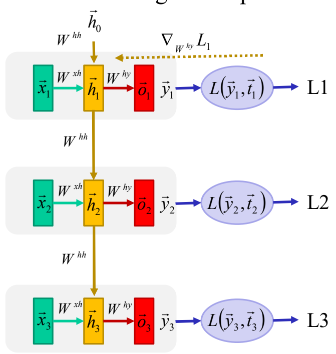


$$\nabla_{W^0} L = \sum_{i=1}^{3} \nabla_{W^0} L_i$$

$$\nabla_{W^1} L = \sum_{i=1}^{3} \nabla_{W^1} L_i$$

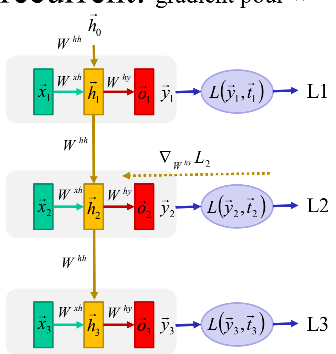
100

Réseau récurrent: gradient pour W^{hy}



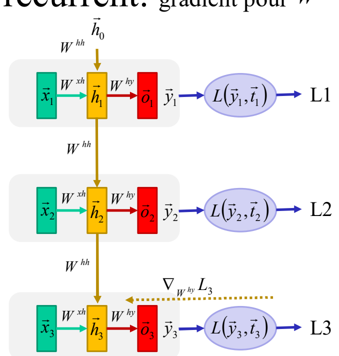
101

Réseau récurrent: gradient pour W^{hy}



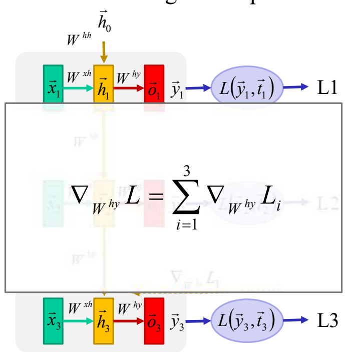
102

Réseau récurrent: gradient pour W^{hy}



103

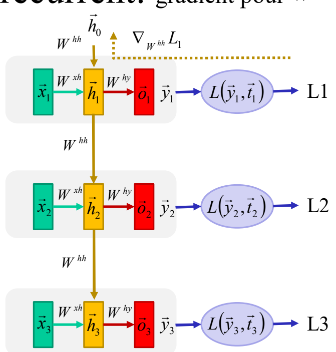
Réseau récurrent: gradient pour W^{hy}



$$\nabla_{W^{hy}} L = \sum_{i=1}^{3} \nabla_{W^{hy}} L_i$$

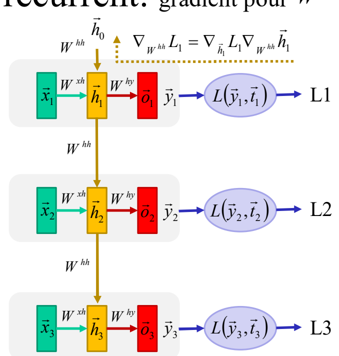
104

Réseau récurrent: gradient pour W^{hh}



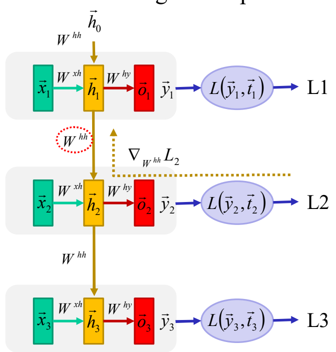
105

Réseau récurrent: gradient pour W^{hh}



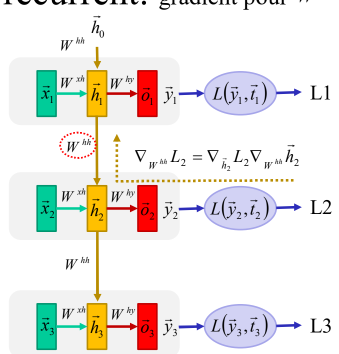
106

Réseau récurrent: gradient pour W^{hh}



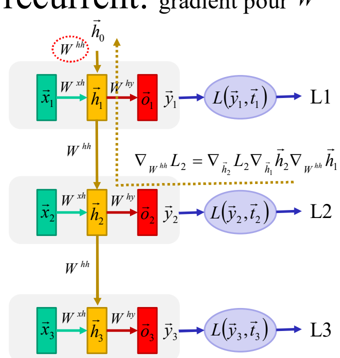
107

Réseau récurrent: gradient pour W^{hh}



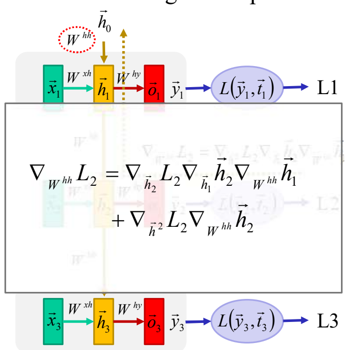
108

Réseau récurrent: gradient pour W^{hh}



109

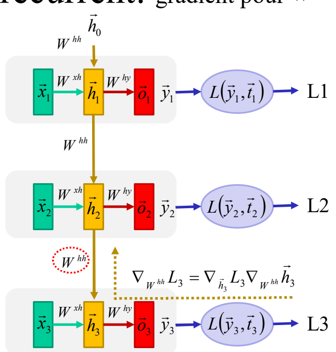
Réseau récurrent: gradient pour W^{hh}



$$\nabla_{W^{hh}} L_2 = \nabla_{\vec{h}_2} L_2 \nabla_{\vec{h}_1} \vec{h}_2 \nabla_{W^{hh}} \vec{h}_1 + \nabla_{\vec{h}^2} L_2 \nabla_{W^{hh}} \vec{h}_2$$

110

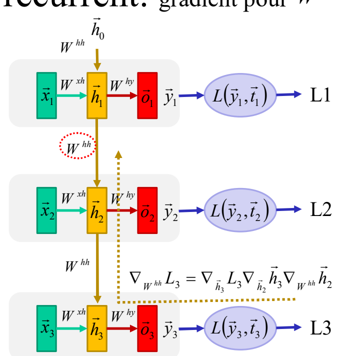
Réseau récurrent: gradient pour W^{hh}



$$\nabla_{W^{hh}} L_3 = \nabla_{\vec{h}_3} L_3 \nabla_{W^{hh}} \vec{h}_3$$

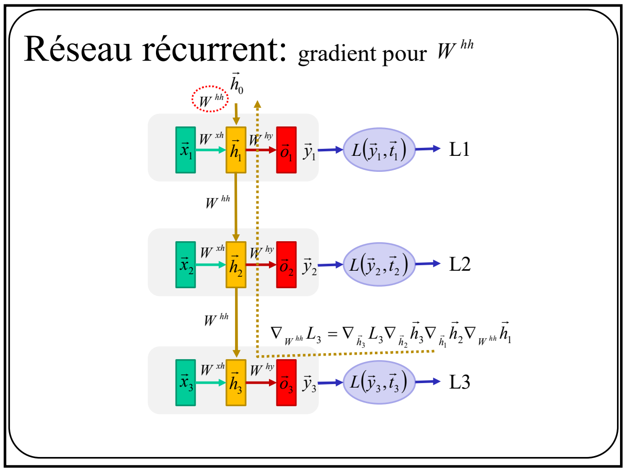
111

Réseau récurrent: gradient pour W^{hh}

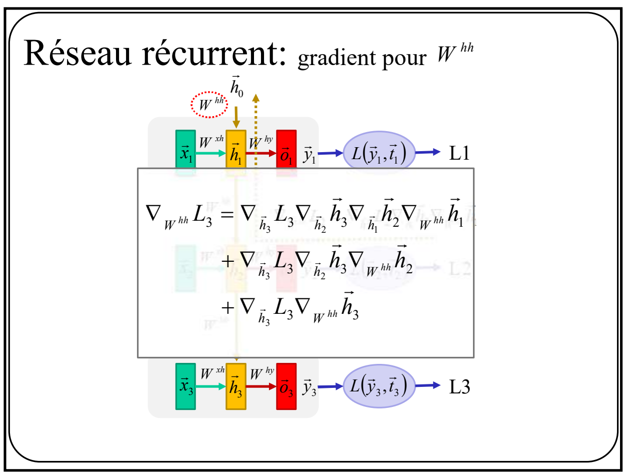


$$\nabla_{W^{hh}} L_3 = \nabla_{\vec{h}_3} L_3 \nabla_{\vec{h}_2} \vec{h}_3 \nabla_{W^{hh}} \vec{h}_2$$

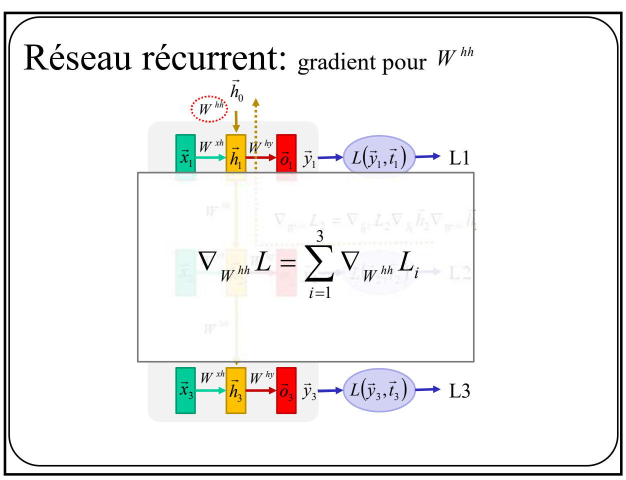
112


Réseau récurrent: gradient pour W^{hh}
$\vec{h}_0$
W^{hh}
W^{xh}
W^{hy}
$L(\vec{y}_1, \vec{t}_1)$
L1
$L(\vec{y}_2, \vec{t}_2)$
L2
$L(\vec{y}_3, \vec{t}_3)$
L3
$\nabla_{W^{hh}} L_3 = \nabla_{\vec{h}_3} L_3 \nabla_{\vec{h}_2} \vec{h}_3 \nabla_{\vec{h}_1} \vec{h}_2 \nabla_{W^{hh}} \vec{h}_1$

113

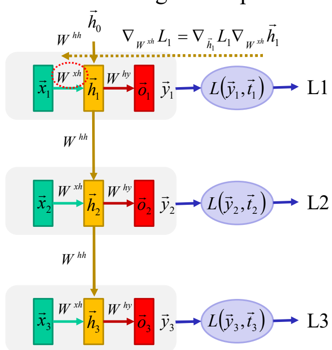

Réseau récurrent: gradient pour W^{hh}
$\vec{h}_0$
W^{hh}
W^{xh}
W^{hy}
$L(\vec{y}_1, \vec{t}_1)$
L1
$\nabla_{W^{hh}} L_3 = \nabla_{\vec{h}_3} L_3 \nabla_{\vec{h}_2} \vec{h}_3 \nabla_{\vec{h}_1} \vec{h}_2 \nabla_{W^{hh}} \vec{h}_1$
$+ \nabla_{\vec{h}_3} L_3 \nabla_{\vec{h}_2} \vec{h}_3 \nabla_{W^{hh}} \vec{h}_2$
$+ \nabla_{\vec{h}_3} L_3 \nabla_{W^{hh}} \vec{h}_3$
$L(\vec{y}_3, \vec{t}_3)$
L3

114


Réseau récurrent: gradient pour W^{hh}
$\vec{h}_0$
W^{hh}
W^{xh}
W^{hy}
$L(\vec{y}_1, \vec{t}_1)$
L1
$\nabla_{W^{hh}} L = \sum_{i=1}^{3} \nabla_{W^{hh}} L_i$
$L(\vec{y}_3, \vec{t}_3)$
L3

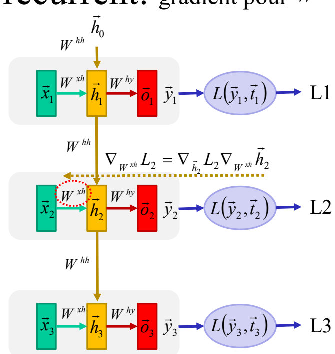
116

Réseau récurrent: gradient pour W^{xh}



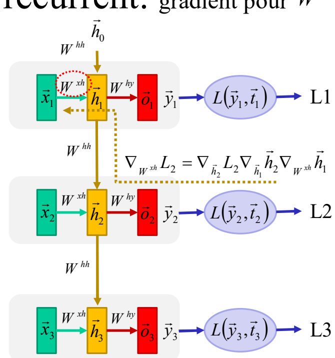
118

Réseau récurrent: gradient pour W^{xh}



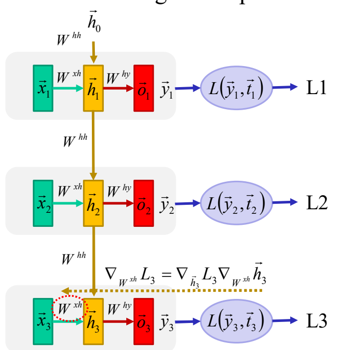
119

Réseau récurrent: gradient pour W^{xh}



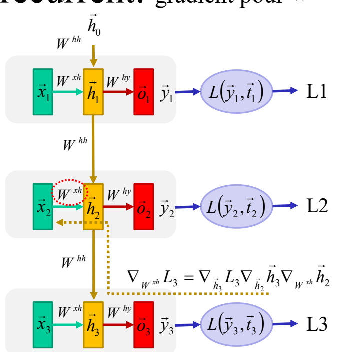
120

Réseau récurrent: gradient pour W^{xh}



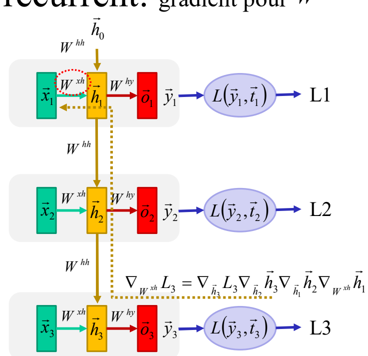
121

Réseau récurrent: gradient pour W^{xh}



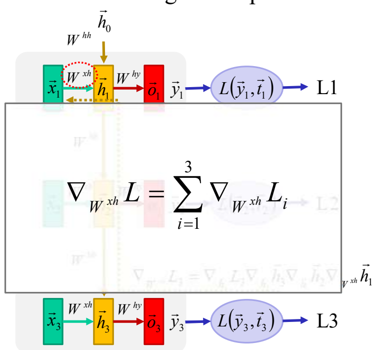
122

Réseau récurrent: gradient pour W^{xh}



123

Réseau récurrent: gradient pour W^{xh}



$$\nabla_{W^{xh}} L = \sum_{i=1}^{3} \nabla_{W^{xh}} L_i$$

124

Réseau récurrent: calcul du gradient

Moins difficile qu'il n'y paraît.

```python
# backward pass: compute gradients going backwards
dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
dbh, dby = np.zeros_like(bh), np.zeros_like(by)
dhnext = np.zeros_like(hs[0])
for t in reversed(xrange(len(inputs))):
  dy = np.copy(ps[t])
  dy[targets[t]] -= 1 # backprop into y. see http://cs231n.github.io/neural-networks-case-study/#grad if confused here
  dWhy += np.dot(dy, hs[t].T)
  dby += dy
  dh = np.dot(Why.T, dy) + dhnext # backprop into h
  dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
  dbh += dhraw
  dWxh += np.dot(dhraw, xs[t].T)
  dWhh += np.dot(dhraw, hs[t-1].T)
  dhnext = np.dot(Whh.T, dhraw)
for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
  np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
return loss, dWxh, dWhh, dWhy, dbh, dby, 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:

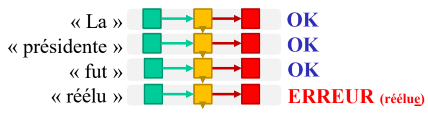
difficile à établir des
relations à longue distance

126

126

Exemples: analyse grammaticale

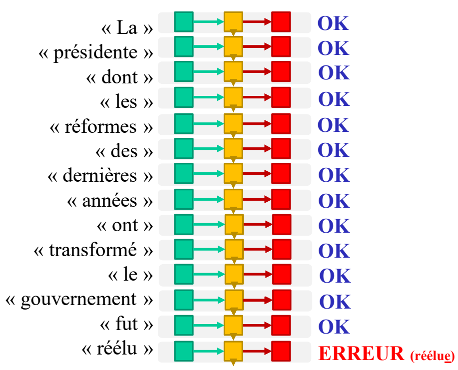
Entraîner un réseau à détecter des erreurs grammaticales



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

127

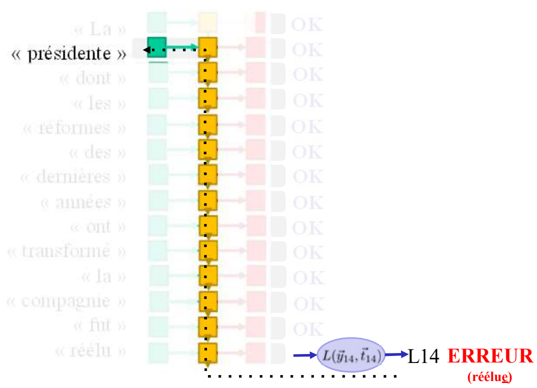
127



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

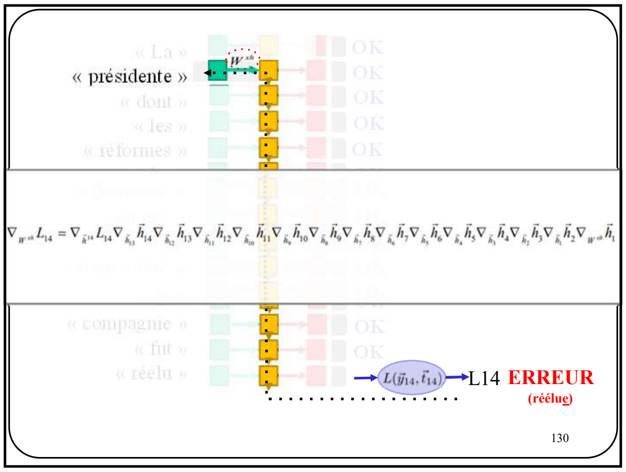
128

128

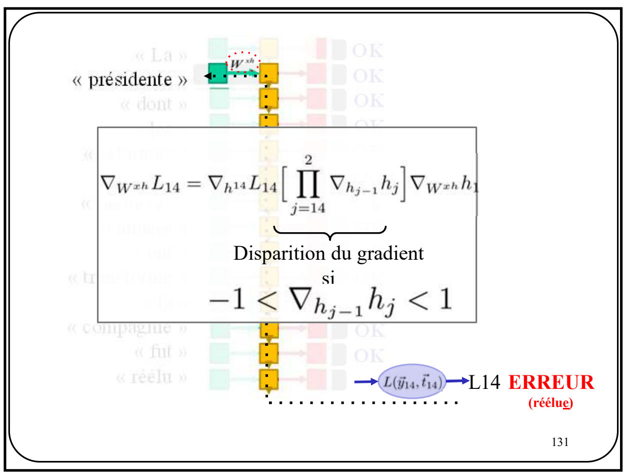


129

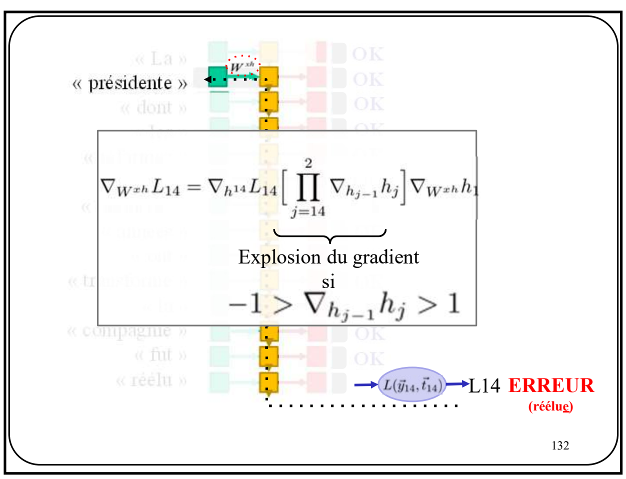
129


« La »
« présidente »
« dont »
« les »
« réformes »
« compagnie »
« fut »
« réélu »
W^{xh}
OK
$\nabla_{W^{xh}} L_{14} = \nabla_{\vec{h}^{14}} L_{14} \nabla_{\vec{h}_{13}} \vec{h}_{14} \nabla_{\vec{h}_{12}} \vec{h}_{13} \nabla_{\vec{h}_{11}} \vec{h}_{12} \nabla_{\vec{h}_{10}} \vec{h}_{11} \nabla_{\vec{h}_{9}} \vec{h}_{10} \nabla_{\vec{h}_{8}} \vec{h}_{9} \nabla_{\vec{h}_{7}} \vec{h}_{8} \nabla_{\vec{h}_{6}} \vec{h}_{7} \nabla_{\vec{h}_{5}} \vec{h}_{6} \nabla_{\vec{h}_{4}} \vec{h}_{5} \nabla_{\vec{h}_{3}} \vec{h}_{4} \nabla_{\vec{h}_{2}} \vec{h}_{3} \nabla_{\vec{h}_{1}} \vec{h}_{2} \nabla_{W^{xh}} \vec{h}_{1}$
$L(\vec{y}_{14}, \vec{t}_{14})$
L14 ERREUR
(réélue)
130

130


« La »
« présidente »
« dont »
« compagnie »
« fut »
« réélu »
W^{xh}
OK
$\nabla_{W^{xh}} L_{14} = \nabla_{h^{14}} L_{14} \Big[\underbrace{\prod_{j=14}^{2} \nabla_{h_{j-1}} h_j}_{\text{Disparition du gradient}} \Big] \nabla_{W^{xh}} h_1$
Disparition du gradient
si
$-1 < \nabla_{h_{j-1}} h_j < 1$
$L(\vec{y}_{14}, \vec{t}_{14})$
L14 ERREUR
(réélue)
131

131


« La »
« présidente »
« dont »
« compagnie »
« fut »
« réélu »
W^{xh}
OK
$\nabla_{W^{xh}} L_{14} = \nabla_{h^{14}} L_{14} \Big[\underbrace{\prod_{j=14}^{2} \nabla_{h_{j-1}} h_j}_{\text{Explosion du gradient}} \Big] \nabla_{W^{xh}} h_1$
Explosion du gradient
si
$-1 > \nabla_{h_{j-1}} h_j > 1$
$L(\vec{y}_{14}, \vec{t}_{14})$
L14 ERREUR
(réélue)
132

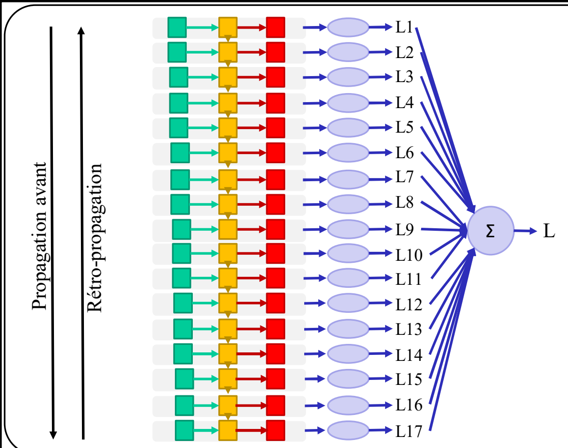
132

Problème connexe

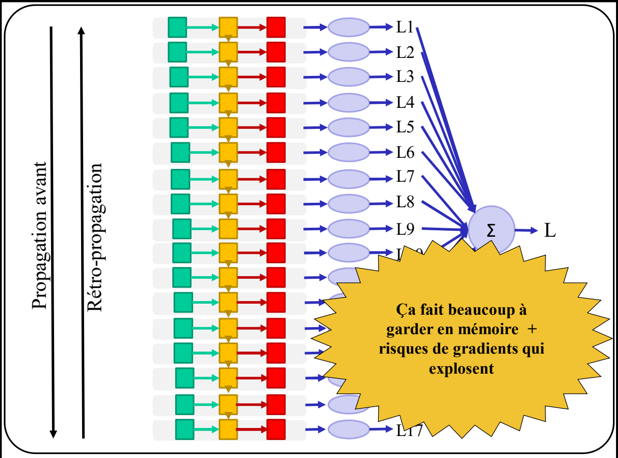
Gestion de la mémoire

133

133



134

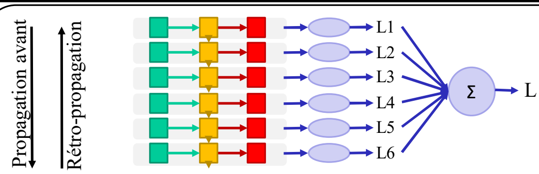


135

Solution pour la gestion de la mémoire
Fenêtres coulissantes

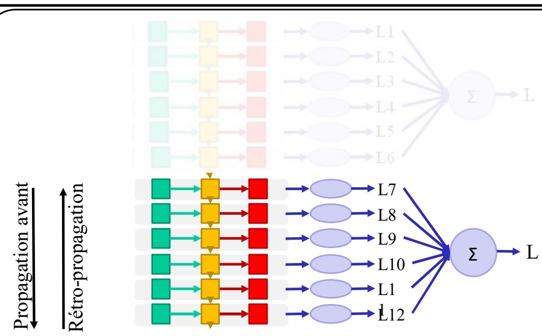
136

136



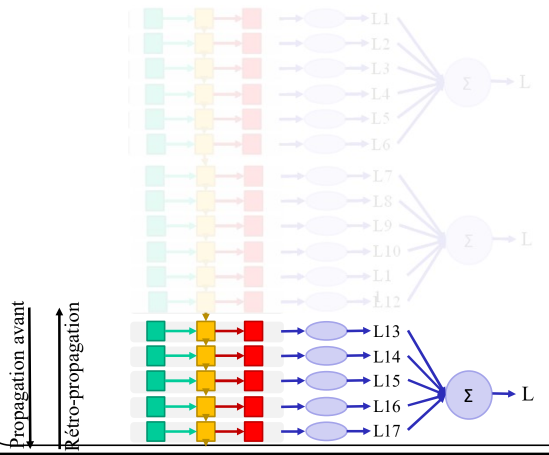
Lorsque les séquences sont trop longues
On entraîne le réseau par fenêtres coulissantes

137



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

138



139

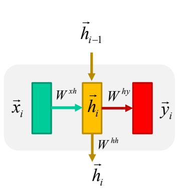
Solution à la disparition du gradient:

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

140

140

Illustration + formulation d'un RNN

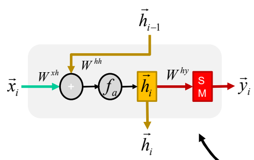


$$\vec{h}_i = f_a\left(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}\right)$$
$$\hat{y}_i = W^{hy}\vec{h}_i$$
$$\vec{y}_i = SMAX\left(\hat{y}_i\right)$$

141

141

Autre illustration du même RNN



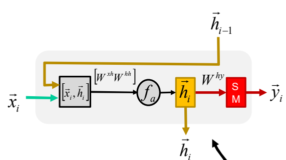
$$\vec{h}_i = f_a\left(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}\right)$$
$$\hat{y}_i = W^{hy}\vec{h}_i$$
$$\vec{y}_i = SMAX(\hat{y}_i)$$

Identique

142

142

Autre illustration du même RNN



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

Encore identique

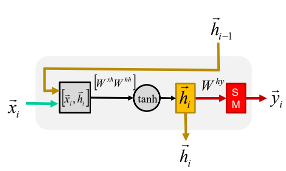
143

143

GRU (Gated Recurrent Unit)

Modif 1

$$f_a = \tanh$$



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

144

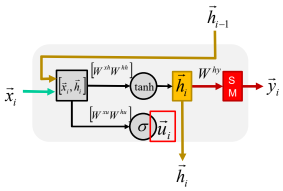
144

GRU (Gated Recurrent Unit)

Modif 2

Update gate

$\sigma = \text{sigmoid}$



$$\boxed{\vec{u}_i = \sigma\left(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}\right)}$$
$$\vec{h}_i = \tanh\left(W^{xh}\vec{x}_i + W^{hh}\vec{h}_{i-1}\right)$$
$$\hat{y}_i = W^{hy}\vec{h}_i$$
$$\bar{y}_i = SMAX\left(\hat{y}_i\right)$$

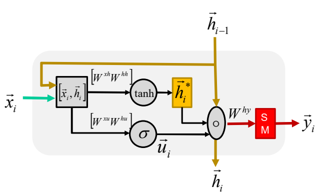
145

145

GRU (Gated Recurrent Unit)

Modif 2

Update gate



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

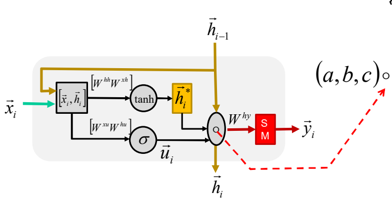
146

146

GRU (Gated Recurrent Unit)

Modif 2

« element-wise product »
ou
« Hadamard product »



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

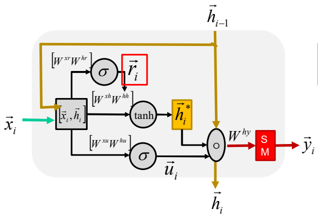
147

147

GRU (Gated Recurrent Unit)

Modif 3

Reset gate



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

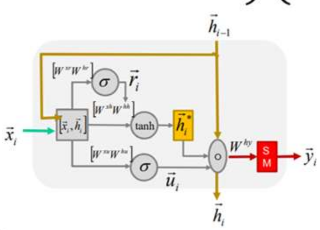
148

148

Comprendre les *gates*

$$\textbf{\textit{SI}} \quad \begin{matrix} u_i & = 1 \\ r_i & = 1 \end{matrix}$$

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



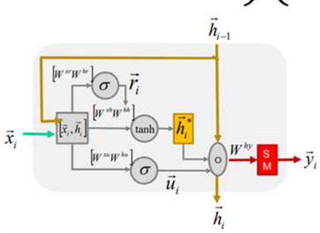
149

149

Comprendre les *gates*

$$\textbf{\textit{SI}} \quad \begin{matrix} u_i & = 1 \\ r_i & = 1 \end{matrix}$$

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



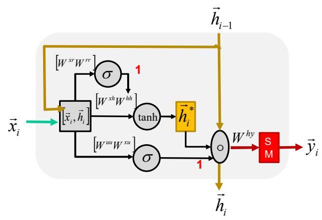
150

150

Comprendre les *gates*

SI $\left.\begin{aligned} u_i &= 1 \\ r_i &= 1 \end{aligned}\right\}\Bigg\{$

$$\vec{u}_i = \sigma\left(W^{xu}\vec{x}_i + W^{hu}\vec{h}_{i-1}\right)$$
$$\vec{r}_i = \cancel{\sigma\left(W^{xr}\vec{x}_i + W^{hr}\vec{h}_{i-1}\right)}\ 1$$
$$\vec{h}_i^* = \tanh\left(W^{xh}\vec{x}_i + W^{hh}\left(\cancel{\vec{r}_i}\,1 \circ \vec{h}_{i-1}\right)\right)$$
$$\vec{h}_i = \cancel{\vec{u}_i}\,1 \circ \vec{h}_i^* + \cancel{\left(1-\vec{u}_i\right)\circ\vec{h}_{i-1}}$$
$$\hat{y}_i = W^{hy}\vec{h}_i$$
$$\vec{y}_i = SMAX\left(\hat{y}_i\right)$$



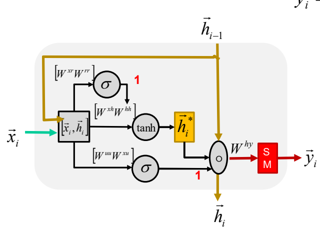
151

151

Comprendre les *gates*

SI $\left.\begin{aligned} \vec{u}_i &= 1 \\ \vec{r}_i &= 1 \end{aligned}\right\}\Bigg\{$

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



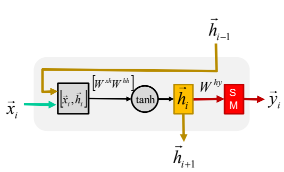
152

152

Comprendre les *gates*

SI $\left.\begin{aligned} \vec{u}_i &= 1 \\ \vec{r}_i &= 1 \end{aligned}\right\}\Bigg\{$

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



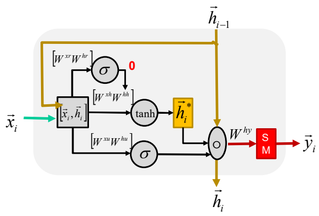
RNN de base!

153

153

SI $\vec{u}_i \neq 1$, $\vec{r}_i = 0$

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



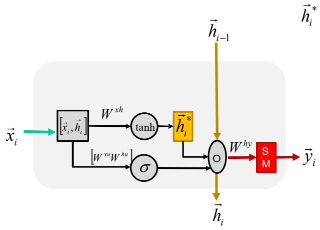
154

154

SI $\vec{u}_i \neq 1$, $\vec{r}_i = 0$

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

$\vec{h}_i^*$ **ne dépend que de** $\vec{x}_i$ **et non de** $\vec{h}_{i-1}$

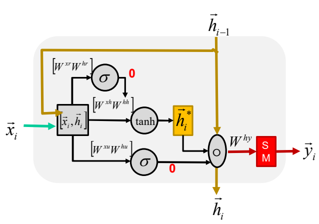


155

155

SI $\vec{u}_i = 0$, $\vec{r}_i = 0$

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



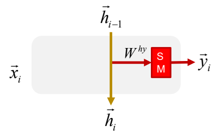
156

156

SI $\begin{matrix}\vec{u}_i = 0 \\ \vec{r}_i = 0\end{matrix} \Bigg\}\Bigg\{ \begin{matrix}\vec{h}_i = \vec{h}_{i-1} \\ \hat{y}_i = W^{cy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i)\end{matrix}$

$\vec{h}_{i-1}$ **est recopié**

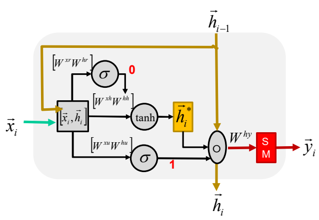
$\vec{x}_i$ **est ignoré**
Aucune disparition de gradient



157

157

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



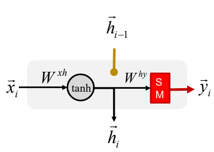
158

158

SI $\begin{matrix}\vec{u}_i = 1 \\ \vec{r}_i = 0\end{matrix} \Bigg\}\Bigg\{ \begin{matrix}\vec{h}_i = \tanh\left(W^{xh}\vec{x}_i\right) \\ \hat{y}_i = W^{hy}\vec{h}_i \\ \vec{y}_i = SMAX(\hat{y}_i)\end{matrix}$

$\vec{h}_{i-1}$ **est ignoté**

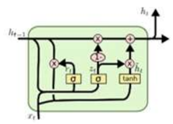
$\vec{x}_i$ **est le seul utilisé**
Gradient temporel <u>bloqué</u>



159

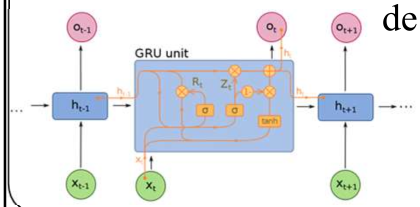
159

Gated Recurrent Unit (GRU)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

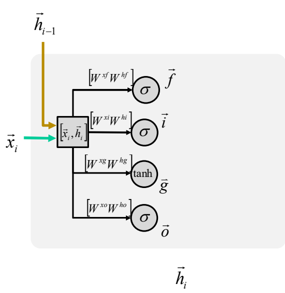
Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes



Le web regorge d'illustrations de GRU!

160

LSTM (Long Short Term Memory)

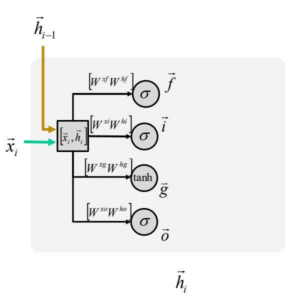


$\vec{f}$: forget gate
$\vec{i}$: input gate
$\vec{g}$: gate gate
$\vec{o}$: output gate

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

161

LSTM (Long Short Term Memory)



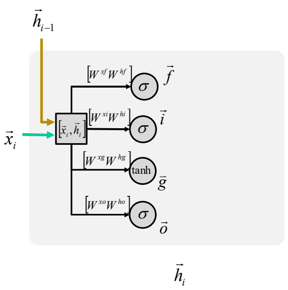
$\vec{f}$: forget gate
$\vec{i}$: input gate
$\vec{g}$: gate gate

Modèle récurrent le plus utilisé. À bien comprendre !

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

162

LSTM (Long Short Term Memory)

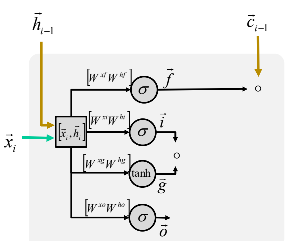


$$\vec{f} = \sigma\left(W^{xf}\vec{x}_i + W^{hf}\vec{h}_{i-1}\right)$$
$$\vec{i} = \sigma\left(W^{xi}\vec{x}_i + W^{hi}\vec{h}_{i-1}\right)$$
$$\vec{g} = \tanh\left(W^{xg}\vec{x}_i + W^{hg}\vec{h}_{i-1}\right)$$
$$\vec{o} = \sigma\left(W^{xo}\vec{x}_i + W^{ho}\vec{h}_{i-1}\right)$$

163

163

LSTM (Long Short Term Memory)

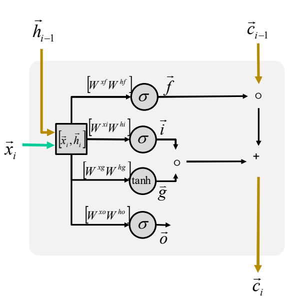


$$\vec{f} \circ \vec{c}_i$$
$$\vec{i} \circ \vec{g}$$

164

164

LSTM (Long Short Term Memory)

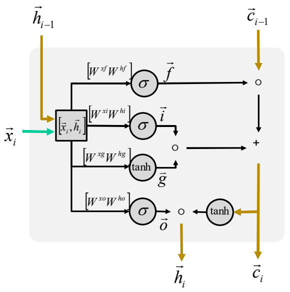


$$\vec{c}_i = \vec{f} \circ \vec{c}_i + \vec{i} \circ \vec{g}$$

165

165

LSTM (Long Short Term Memory)

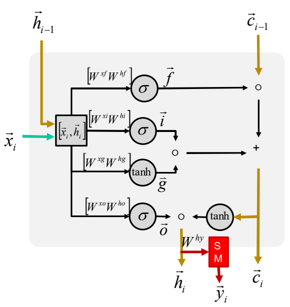


$$\vec{c}_i = \vec{f} \circ \vec{c}_i + \vec{i} \circ \vec{g}$$
$$\vec{h}_i = \vec{o} \circ \tanh(\vec{c}_i)$$

166

166

LSTM (Long Short Term Memory)

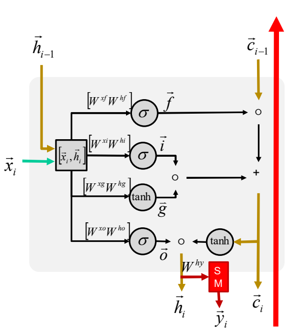


$$\vec{c}_i = \vec{f} \circ \vec{c}_i + \vec{i} \circ \vec{g}$$
$$\vec{h}_i = \vec{o} \circ \tanh(\vec{c}_i)$$
$$\vec{y}_i = SMAX\left(W^{hy}\vec{h}_i\right)$$

167

167

LSTM (Long Short Term Memory)

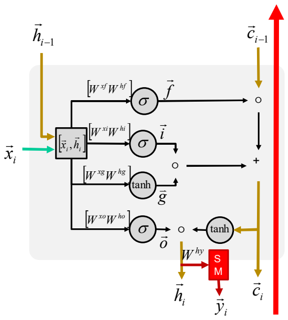


Gradient simplifié

168

168

LSTM (Long Short Term Memory)



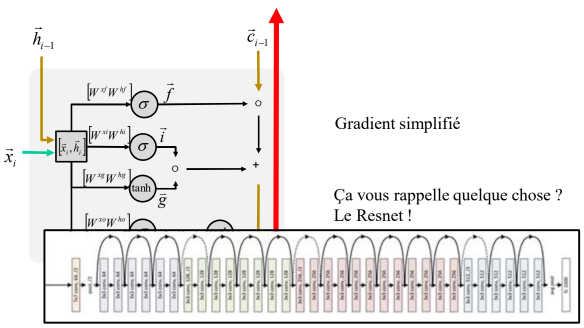
Gradient simplifié

Ça vous rappelle quelque chose ?
Le Resnet !

169

169

LSTM (Long Short Term Memory)



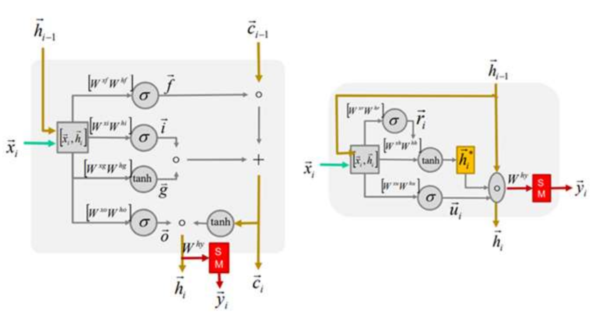
Gradient simplifié

Ça vous rappelle quelque chose ?
Le Resnet !

ResNET

170

LSTM et GRU

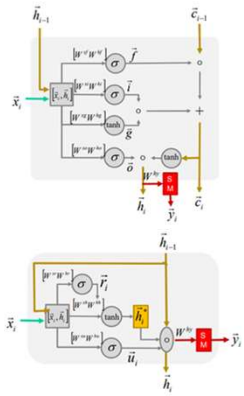


171

171

LSTM et GRU

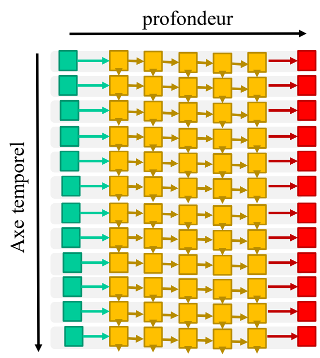
- Servent à protéger le gradient
- Conçus empiriquement
- GRU légèrement plus simple
- Les “gates” ne servent qu’à bloquer ou permettre à l’information (données ou temporelle) de passer



172

172

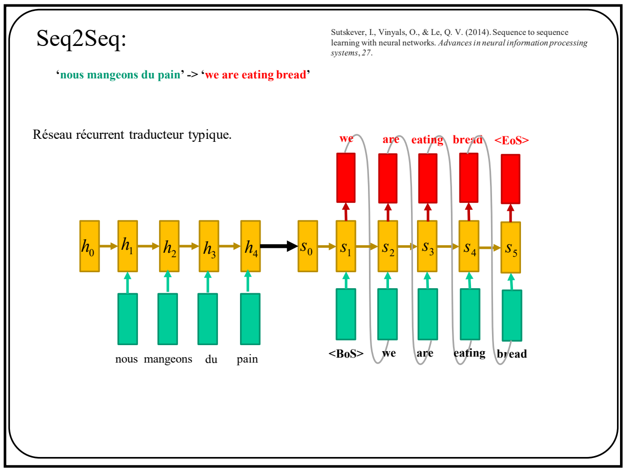
RNN multi-couches



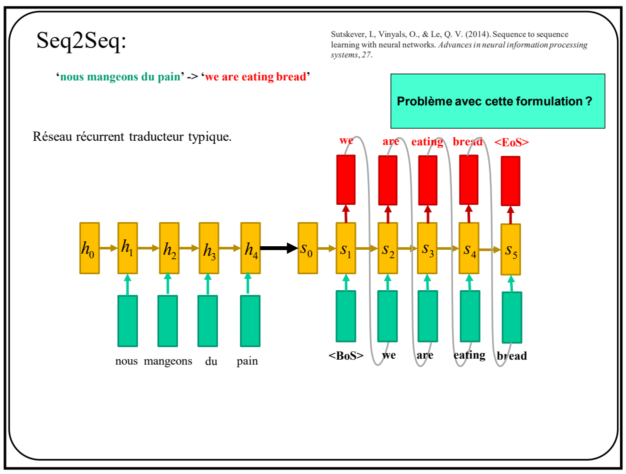
173

Modèles d’attention

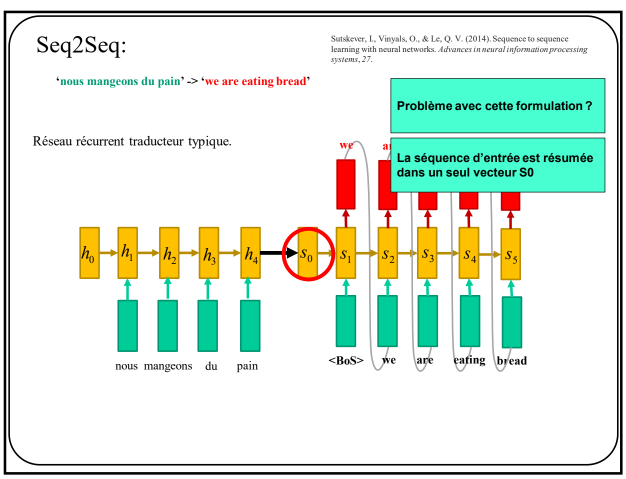
174


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.
we are eating bread <EoS>
nous mangeons du pain
<BoS> we are eating bread

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.
we are eating bread <EoS>
nous mangeons du pain
<BoS> we are eating bread

176


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.
La séquence d'entrée est résumée dans un seul vecteur S0
nous mangeons du pain
<BoS> we are eating bread

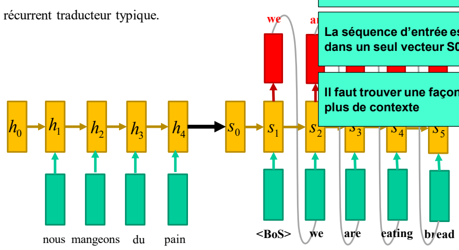
177

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.



Problème avec cette formulation ?

La séquence d'entrée est résumée dans un seul vecteur S0

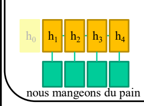
Il faut trouver une façon d'ajouter plus de contexte

178

Seq2Seq avec attention
(version simplifiée)

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

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



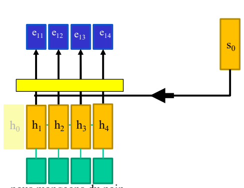
s_0 = init value

179

Seq2Seq avec attention
(version simplifiée)

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

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i \qquad \in R$$

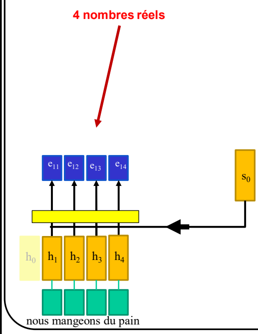


180

Seq2Seq avec attention
(version simplifiée)

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

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i \in R$$



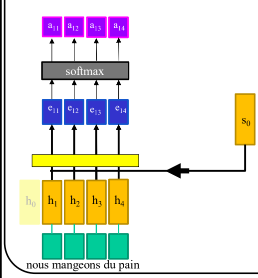
181

Seq2Seq avec attention
(version simplifiée)

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

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i \in R$$

$$a_{1i} = \frac{\exp(e_{1i})}{\sum_{t=1}^{4}\exp(e_{1t})} \quad \in R$$



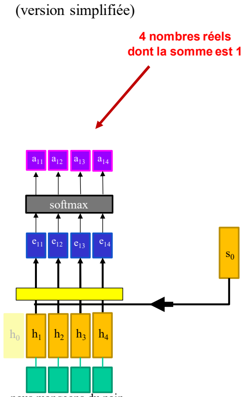
182

Seq2Seq avec attention
(version simplifiée)

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

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i \in R$$

$$a_{1i} = \frac{\exp(e_{1i})}{\sum_{t=1}^{4}\exp(e_{1t})} \quad \in R$$



183

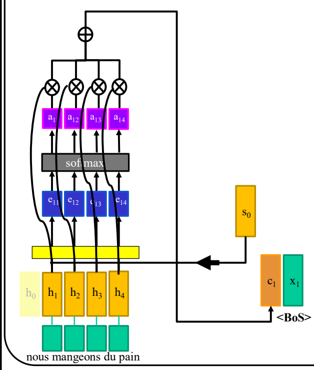
Seq2Seq avec attention
(version simplifiée)

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

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i \in R$$

$$a_{1i} = \frac{\exp(e_{1i})}{\sum_{t=1}^{4}\exp(e_{1t})} \quad \in R$$

$$\vec{c}_1 = \sum_{t=1}^{4} a_{1t}\vec{h}_t \quad \in R^V$$



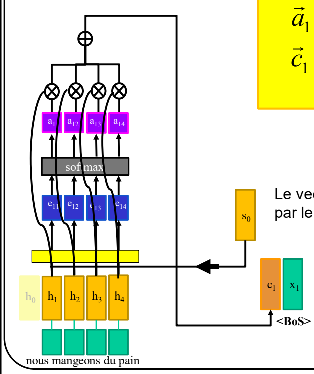
184

Seq2Seq avec attention
(version simplifiée)

$\vec{a}_1$: vecteur d'attention
$\vec{c}_1$: vecteur de context

$$\vec{c}_1 = \sum_{t=1}^{4} a_{1t}\vec{h}_t$$

Le vecteur de contexte est la somme pondérée par le vecteur d'attention des représentations hi



185

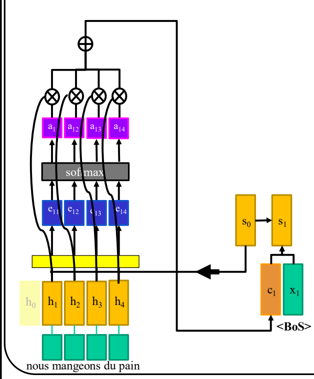
Seq2Seq avec attention
(version simplifiée)

$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i$$

$$a_{1i} = \frac{\exp(e_{1i})}{\sum_{t=1}^{4}\exp(e_{1t})}$$

$$\vec{c}_1 = \sum_{t=1}^{4} a_{1t}\vec{h}_t$$

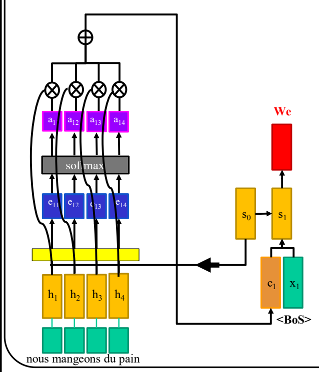
$$s_1 = \tanh\left(W^{ss}\vec{s}_0 + W^{cs}\vec{c}_1 + W^{xs}\vec{x}_1\right)$$



186

Seq2Seq avec attention
(version simplifiée)

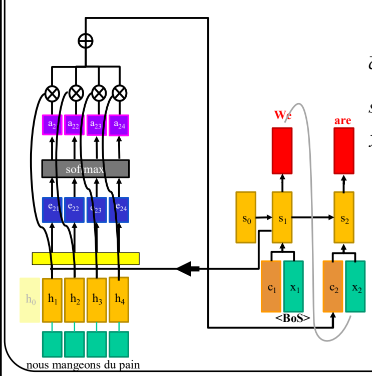
$$e_{1i} = \vec{w}^{sa^T}\vec{s}_0 + \vec{w}^{ha^T}\vec{h}_i$$
$$a_{1i} = \frac{\exp(e_{1i})}{\sum_{t=1}^{4}\exp(e_{1t})}$$
$$\vec{c}_1 = \sum_{t=1}^{4} a_{1t}\vec{h}_t$$
$$s_1 = \tanh\left(W^{ss}\vec{s}_0 + W^{cs}\vec{c}_1 + W^{xs}\vec{x}_1\right)$$
$$y_1 = \text{softmax}\left(W^{sy}\vec{s}_1\right)$$



187

Seq2Seq avec attention
(version simplifiée)

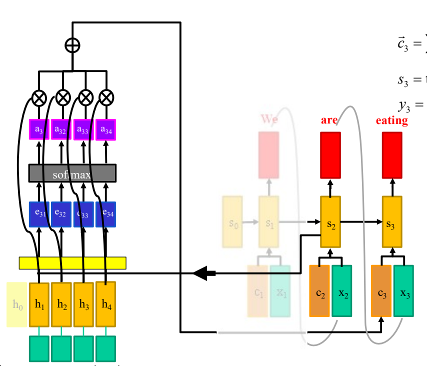
$$e_{2i} = \vec{w}^{sa^T}\vec{s}_1 + \vec{w}^{ha^T}\vec{h}_i$$
$$a_{2i} = \frac{\exp(e_{2i})}{\sum_{t=1}^{4}\exp(e_{2t})}$$
$$\vec{c}_2 = \sum_{t=1}^{4} a_{2t}\vec{h}_t$$
$$s_2 = \tanh\left(W^{ss}\vec{s}_1 + W^{cs}\vec{c}_2 + W^{xs}\vec{x}_2\right)$$
$$y_2 = \text{softmax}\left(W^{sy}\vec{s}_2\right)$$



188

Seq2Seq avec attention
(version simplifiée)

$$e_{3i} = \vec{w}^{sa^T}\vec{s}_2 + \vec{w}^{ha^T}\vec{h}_i$$
$$a_{3i} = \frac{\exp(e_{3i})}{\sum_{t=1}^{4}\exp(e_{3t})}$$
$$\vec{c}_3 = \sum_{t=1}^{4} a_{3t}\vec{h}_t$$
$$s_3 = \tanh\left(W^{ss}\vec{s}_2 + W^{cs}\vec{c}_3 + W^{xs}\vec{x}_3\right)$$
$$y_3 = \text{softmax}\left(W^{sy}\vec{s}_3\right)$$



189

Seq2Seq avec attention
(version simplifiée)

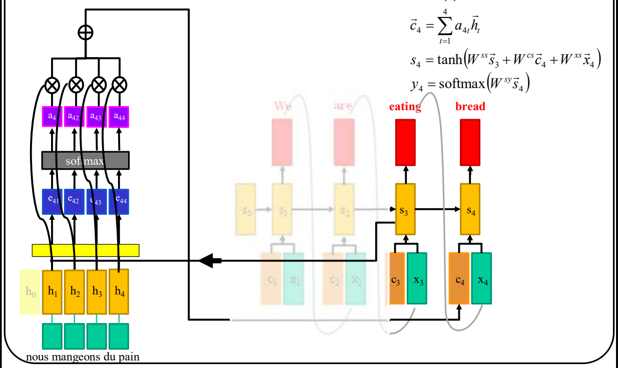
$$e_{4i} = \vec{w}^{sa^T}\vec{s}_3 + \vec{w}^{ha^T}\vec{h}_i$$

$$a_{4i} = \frac{\exp(e_{4i})}{\sum_{t=1}^{4}\exp(e_{4t})}$$

$$\vec{c}_4 = \sum_{t=1}^{4} a_{4t}\vec{h}_t$$

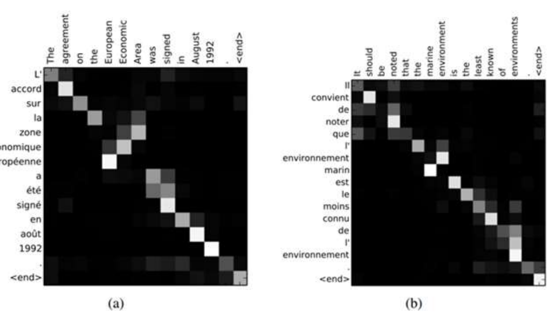
$$s_4 = \tanh\left(W^{ss}\vec{s}_3 + W^{cs}\vec{c}_4 + W^{xs}\vec{x}_4\right)$$

$$y_4 = \mathrm{softmax}\left(W^{sy}\vec{s}_4\right)$$



190

Un vecteur d'attention pour chaque mot

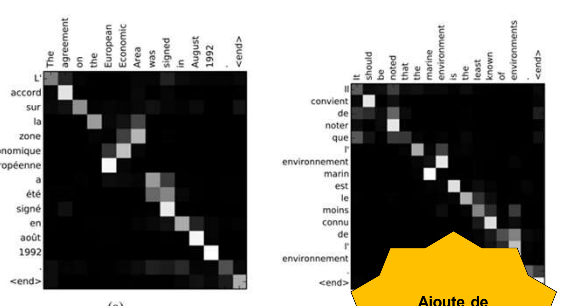


(a) (b)

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)

Revenons à la base : **multiplication matricielle**

Considérons les 4 matrices suivantes

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

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

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

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

Revenons à la base : **multiplication matricielle**

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

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

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

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

Leur multiplication donne:

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

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

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

Auto attention

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

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

Nous mangeons du pain

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

207

207

Auto attention

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

Nous mangeons du pain

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

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

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

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

208

208

Auto attention

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

Nous mangeons du pain

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

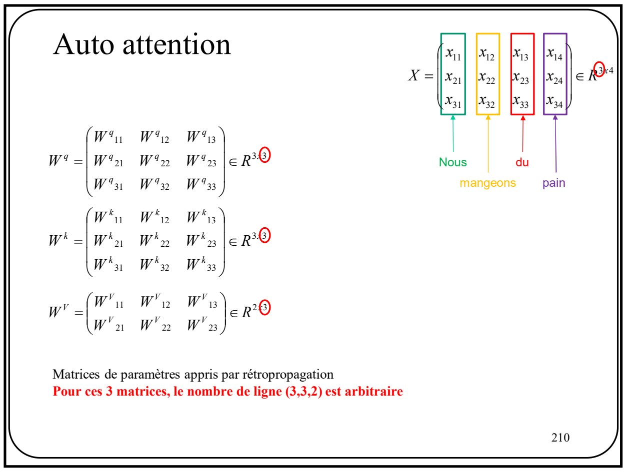
$$W^k = \begin{pmatrix} W^k{}_{11} & W^k{}_{12} & W^k{}_{13} \\ W^k{}_{21} & W^k{}_{22} & W^k{}_{23} \\ W^k{}_{31} & W^k{}_{32} & W^k{}_{33} \end{pmatrix} \in R^{3x3}$$

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

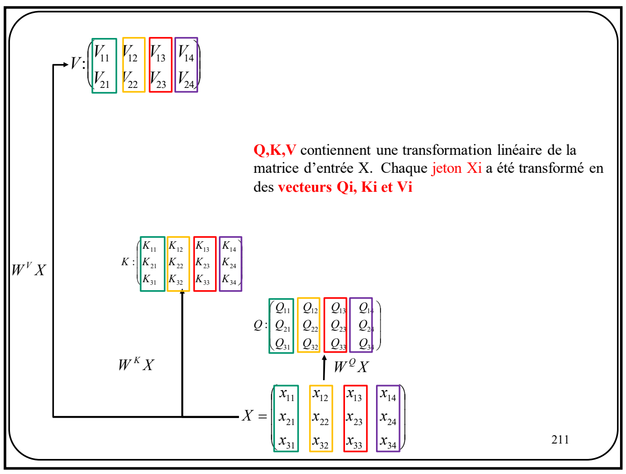
Matrices de paramètres appris par rétropropagation
Pour ces 3 matrices, le nombre de colonnes (3) doit être égale au nombre de lignes dans X (3)

209

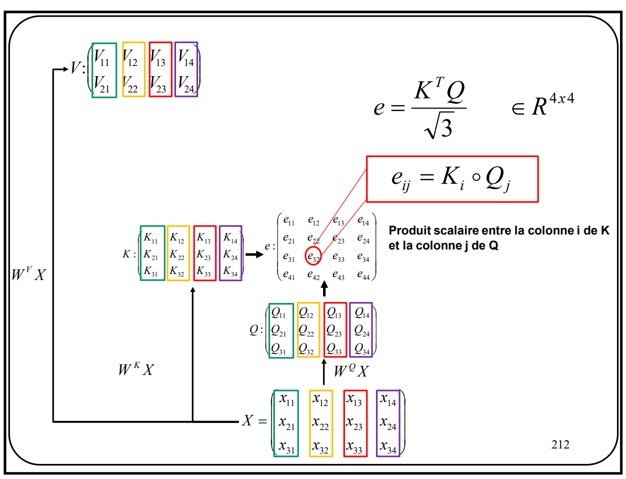
209


Auto attention
$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix} \in R^{3x4}$
Nous
mangeons
du
pain
$W^q = \begin{pmatrix} W^q_{11} & W^q_{12} & W^q_{13} \\ W^q_{21} & W^q_{22} & W^q_{23} \\ W^q_{31} & W^q_{32} & W^q_{33} \end{pmatrix} \in R^{3x3}$
$W^k = \begin{pmatrix} W^k_{11} & W^k_{12} & W^k_{13} \\ W^k_{21} & W^k_{22} & W^k_{23} \\ W^k_{31} & W^k_{32} & W^k_{33} \end{pmatrix} \in R^{3x3}$
$W^V = \begin{pmatrix} W^V_{11} & W^V_{12} & W^V_{13} \\ W^V_{21} & W^V_{22} & W^V_{23} \end{pmatrix} \in R^{2x3}$
Matrices de paramètres appris par rétropropagation
Pour ces 3 matrices, le nombre de ligne (3,3,2) est arbitraire
210

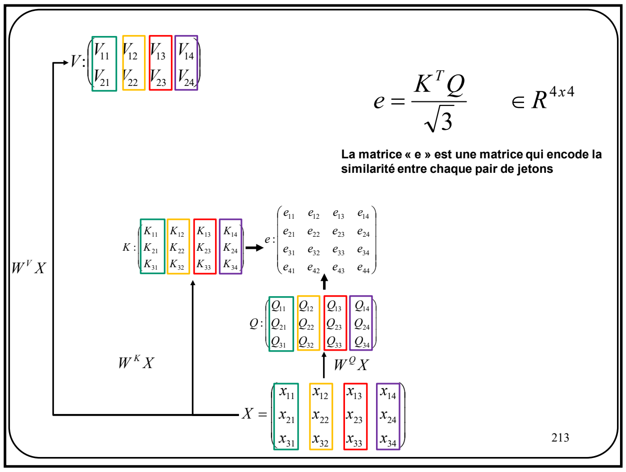
210


$V: \begin{pmatrix} V_{11} & V_{12} & V_{13} & V_{14} \\ V_{21} & V_{22} & V_{23} & V_{24} \end{pmatrix}$
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
$W^V X$
$K: \begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \end{pmatrix}$
$Q: \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \end{pmatrix}$
$W^K X$
$W^Q X$
$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix}$
211

211

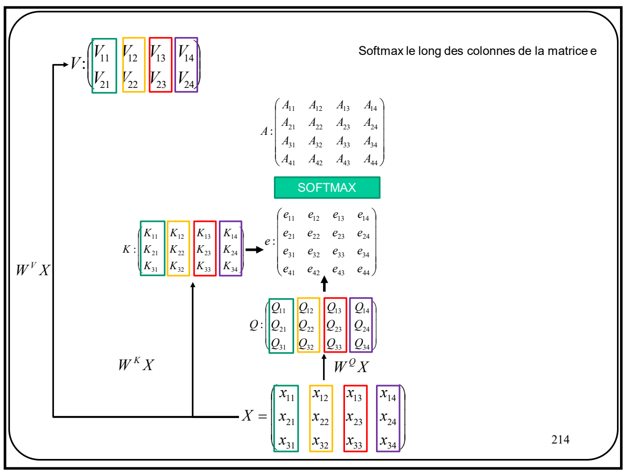

$V: \begin{pmatrix} V_{11} & V_{12} & V_{13} & V_{14} \\ V_{21} & V_{22} & V_{23} & V_{24} \end{pmatrix}$
$e = \frac{K^T Q}{\sqrt{3}} \in R^{4x4}$
$e_{ij} = K_i \circ Q_j$
$e: \begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ e_{41} & e_{42} & e_{43} & e_{44} \end{pmatrix}$
Produit scalaire entre la colonne i de K et la colonne j de Q
$W^V X$
$K: \begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \end{pmatrix}$
$Q: \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \end{pmatrix}$
$W^K X$
$W^Q X$
$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{pmatrix}$
212

212



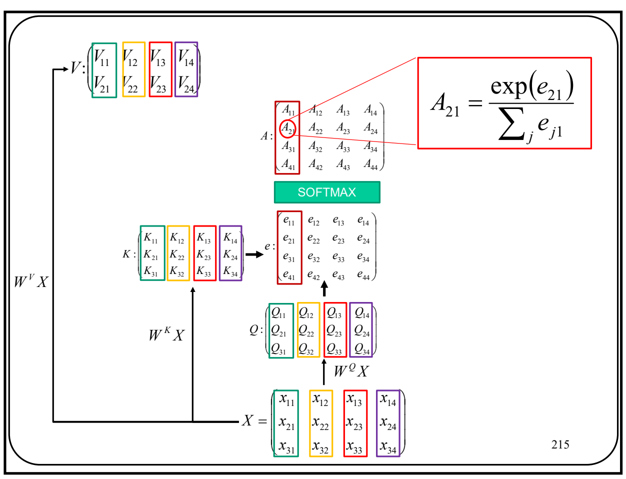
$e = \frac{K^T Q}{\sqrt{3}} \quad \in R^{4x4}$
La matrice « e » est une matrice qui encode la similarité entre chaque pair de jetons
$W^V X$
$W^K X$
$W^Q X$
213

213



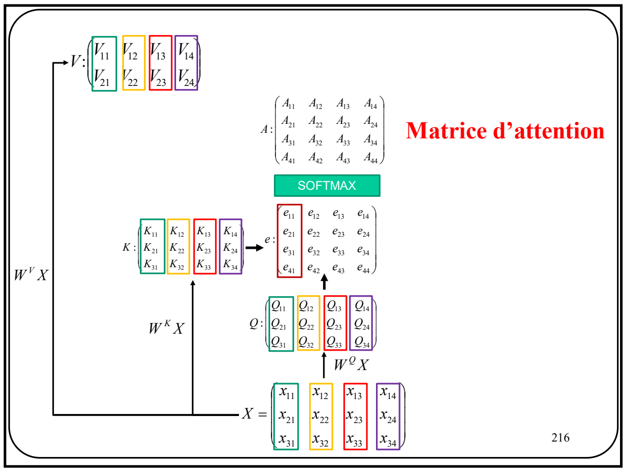
Softmax le long des colonnes de la matrice e
SOFTMAX
$W^V X$
$W^K X$
$W^Q X$
214

214

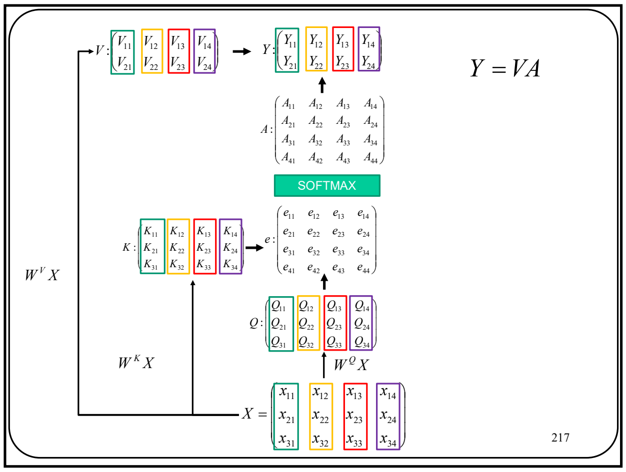


$A_{21} = \frac{\exp(e_{21})}{\sum_j e_{j1}}$
SOFTMAX
$W^V X$
$W^K X$
$W^Q X$
215

215

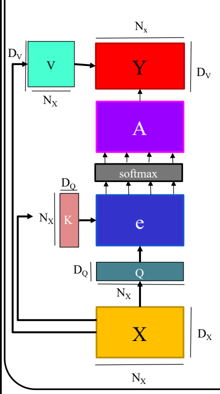


216



217

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



X données
Query: que chercher dans nos données
Key: la "clé" des données
Value: la "valeur" des données
Entries: le "contenu" des données
Attention: la pertinence des entrées

$$Q = W^Q X$$
$$K = W^K X$$
$$V = W^V X$$
$$e = \frac{K^T Q}{\sqrt{D_Q}}$$
$$A_{ji} = \frac{\exp(e_{ji})}{\sum_j e_{ji}}$$
$$Y = VA$$

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.*

218

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

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

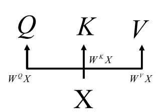
X

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*.

219

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

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

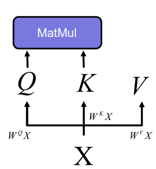


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

220

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

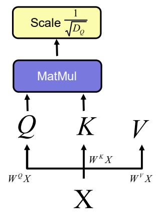
$$Attention(Q,K,V) = V \cdot \text{softmax}\left(\frac{K^T Q}{\sqrt{D_Q}}\right)$$



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

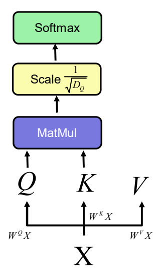


$$Attention(Q, K, V) = V \cdot \mathrm{softmax}\left(\frac{K^T Q}{\sqrt{D_Q}}\right)$$

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

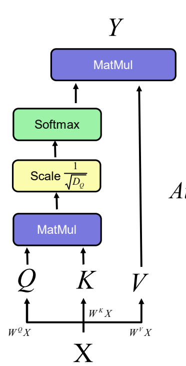


$$Attention(Q, K, V) = V \cdot \mathrm{softmax}\left(\frac{K^T Q}{\sqrt{D_Q}}\right)$$

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



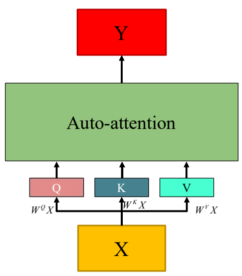
$$Attention(Q, K, V) = V \cdot \mathrm{softmax}\left(\frac{K^T Q}{\sqrt{D_Q}}\right)$$

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)*

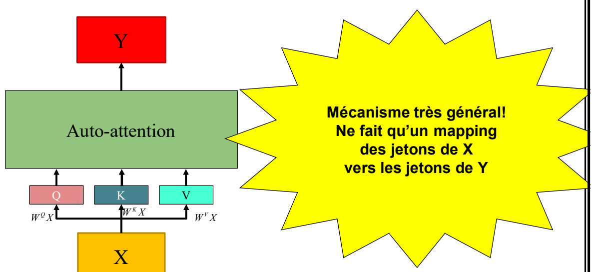
Autre représentation schématique



225

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

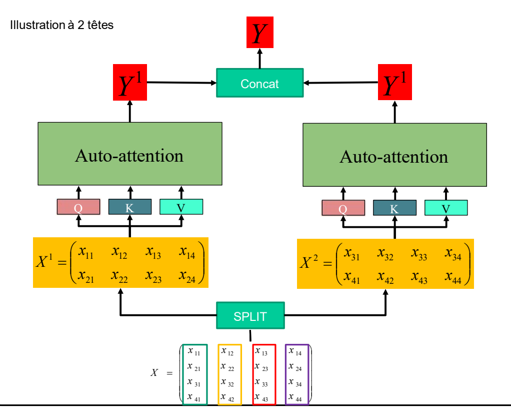
Autre représentation schématique



226

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

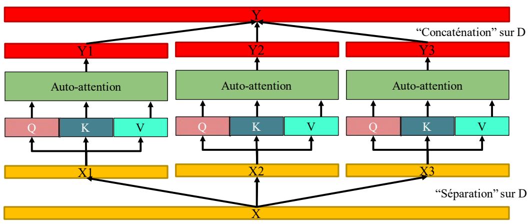
Illustration à 2 têtes



227

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

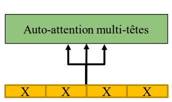
Transformer (Attention is all you need)

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

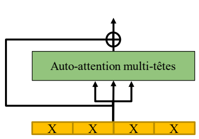


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30.*

232

Transformer (Attention is all you need)

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

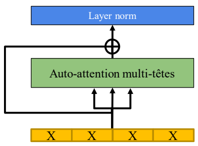


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30.*

233

Transformer (Attention is all you need)

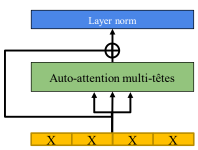
- Auto-attention multi-têtes sur les dimensions de X
- "+ " = connexion résiduelle
- "Layer-norm"

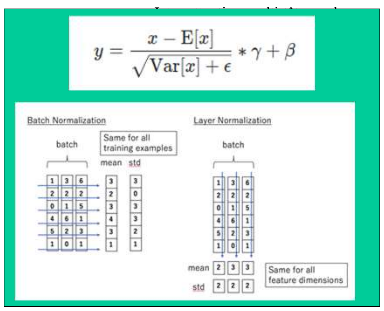


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30.*

234

Transformer (Attention is all you need)





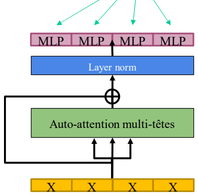
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30.*

235

Transformer (Attention is all you need)

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

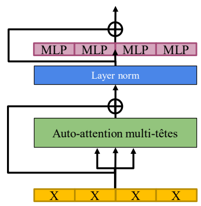
$$mlp(x) = \max(0, xW^1)W^2$$



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)

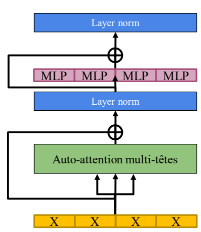


- Intra-attention multi-tête sur les dimensions de X
- "+ " = connexion résiduelle
- "Layer-norm"
- MLP pour chaque jeton
- 2e "+ " = connexion résiduelle

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)

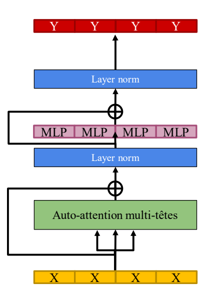


- Auto-attention multi-têtes sur les dimensions de X
- "+ " = connexion résiduelle
- "Layer-norm"
- MLP pour chaque jeton
- 2e "+ " = connexion résiduelle
- 2e "Layer-norm"

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)

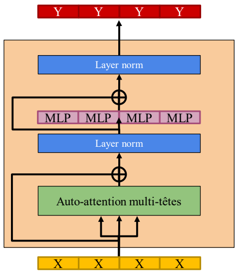


- Auto-attention multi-têtes sur les dimensions de X
- "+ " = connexion résiduelle
- "Layer-norm"
- MLP pour chaque jeton
- 2e "+ " = connexion résiduelle
- 2e "Layer-norm"

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

Transformer (Attention is all you need)

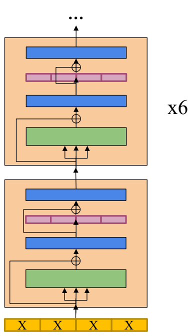


- Auto-attention multi-têtes sur les dimensions de X
- "+ " = connexion résiduelle
- "Layer-norm"
- MLP pour chaque jeton
- 2e "+ " = connexion résiduelle
- 2e "Layer-norm"

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.*

240

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"

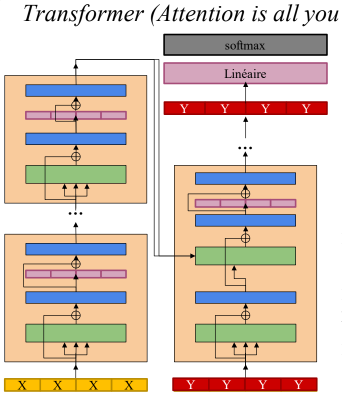
Le bloc est répété 6 fois pour l'encodeur

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.*

241

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.*



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

Le bloc est répété 6 fois pour l'encodeur

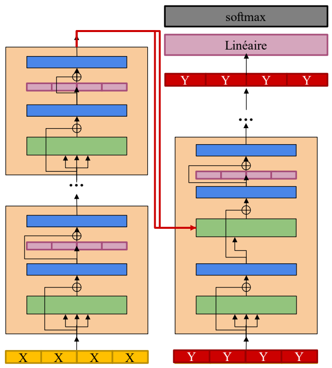
La sortie de l'encodeur sert de K,V pour tous les décodeurs

Le décodeur est répété 6 fois

242

Transformer (Attention is all you need)

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



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

Le bloc est répété 6 fois pour l'encodeur

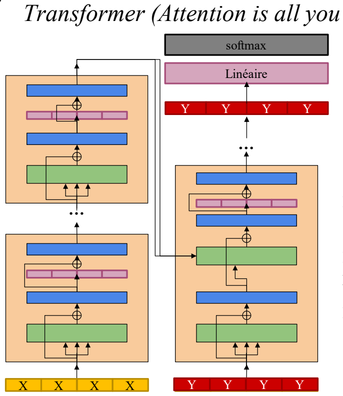
La sortie de l'encodeur sert de K,V pour tous les décodeurs

Le décodeur est répété 6 fois

243

Transformer (Attention is all you need)

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



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

Le bloc est répété 6 fois pour l'encodeur

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

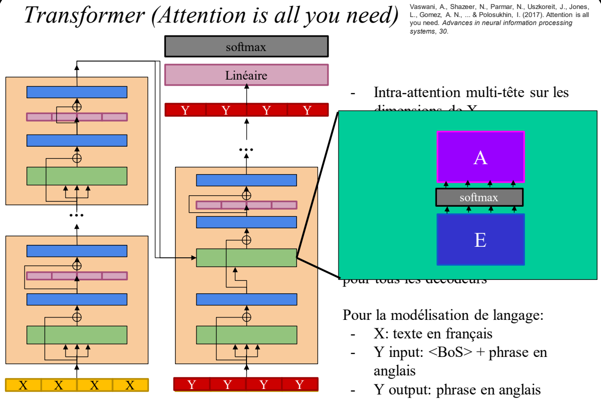
Pour la modélisation de langage:
- X: texte en français
- Y input: <BoS> + phrase en anglais
- Y output: phrase en anglais

244

Transformer (Attention is all you need)

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

- Intra-attention multi-tête sur les

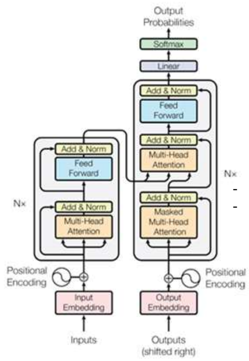


Pour la modélisation de langage:
- X: texte en français
- Y input: <BoS> + phrase en anglais
- Y output: phrase en anglais

245

Transformer (Attention is all you need)

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



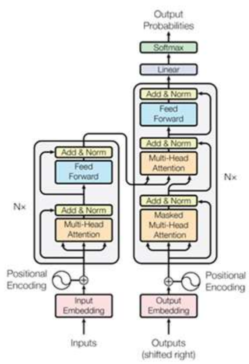
Supers bons tutoriels:

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

249

Transformer (Attention is all you need)

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



Très gourmand en mémoire:

- Matrices de poids W^Q,W^K,W^V
- Couches pleinement connectées
- "Répétition" de paramètres par les multi-têtes

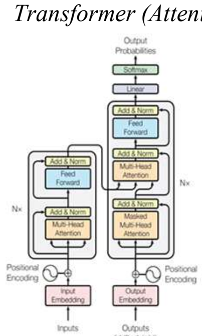
Très demandant en opérations

- Complexité quadratique

250

Transformer (Attention is all you need)

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



Très gourmand en mémoire:

- Matrices de poids W^Q,W^K,W^V
- Couches pleinement connectées
- "Répétition" de paramètres par les multi-têtes

Aussi très populaires !

[PDF] **Attention** is **all you need**
A Vaswani - Advances in Neural Information Processing Systems, 2017 - user.phil.hhu.de
Attention is **all you need Attention** is **all you need** ...
Save Cite Cited by 150580 Related articles

251

Différentes version de transformers

Pour rappel: Resnet-50: 23M de paramètres

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

crédit: Justin Johnson

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	

crédit: Justin Johnson

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)

crédit: Justin Johnson

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	

crédit: Justin Johnson

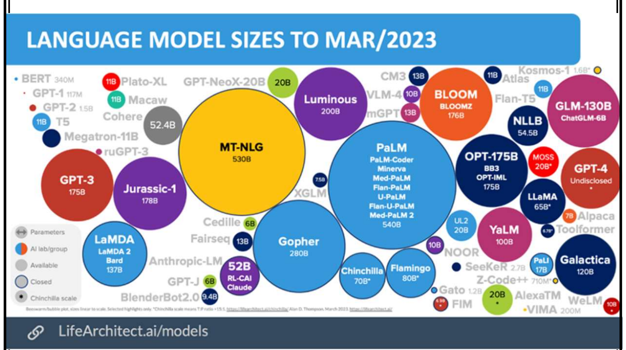
255

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)

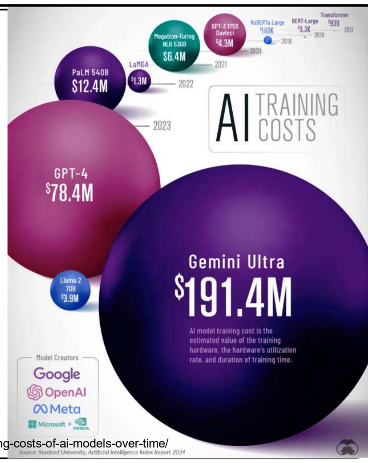
crédit: Justin Johnson

256



https://blogs.cfainstitute.org/investor/2023/05/26/chatgpt-and-large-language-models-six-evolutionary-steps/

257

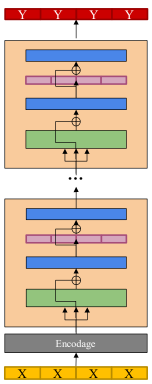


https://www.visualcapitalist.com/training-costs-of-ai-models-over-time/

258

GPT1-2-3

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



Generative-Pre-Training

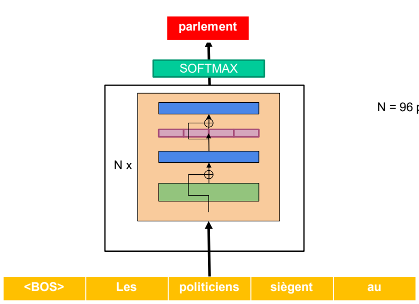
GTP 1,2,3 entraînent l'encodeur du transformer sur des données non annotée (d'où le « pre-training »)

GPT3 a 96 blocks d'auto-attention!

259

GPT1-2-3

Pré-entraînement : le but est d'entraîner un réseau **à prédire le prochain mot**.



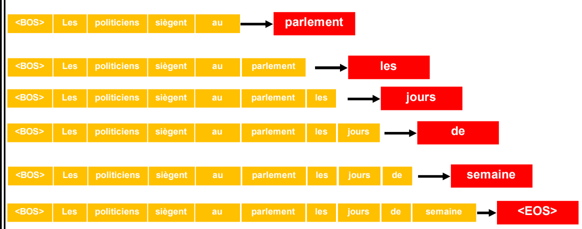
N = 96 pour GPT3

260

260

GPT1-2-3

Et on recommence:



261

261

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

262

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

263

GPT-1-2-3

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

GPT-2

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

Table 2. Architecture hyperparameters for the 4 model sizes.

GPT-3

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

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

264

GPT1-2-3

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

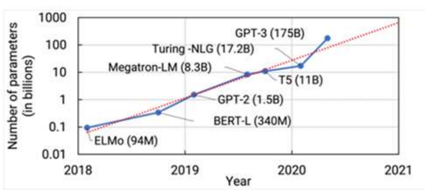


Figure 1. Trend of state-of-the-art NLP model sizes with time.

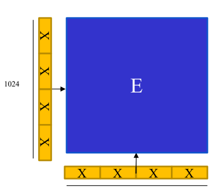
https://developer.nvidia.com/blog/scaling-language-model-training-to-a-trillion-parameters-using-megatron/

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

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

265

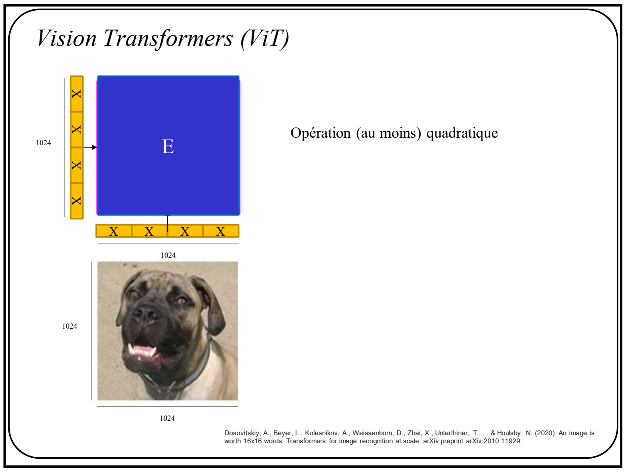
Vision Transformers (ViT)



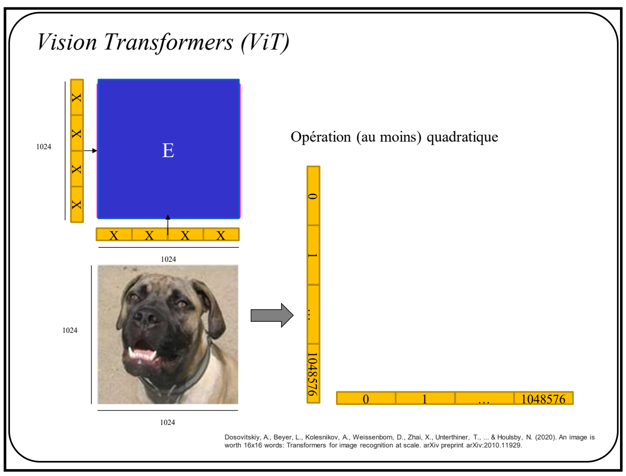
Opération (au moins) quadratique = 1024^2 opérations et composantes à garder en mémoire

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.

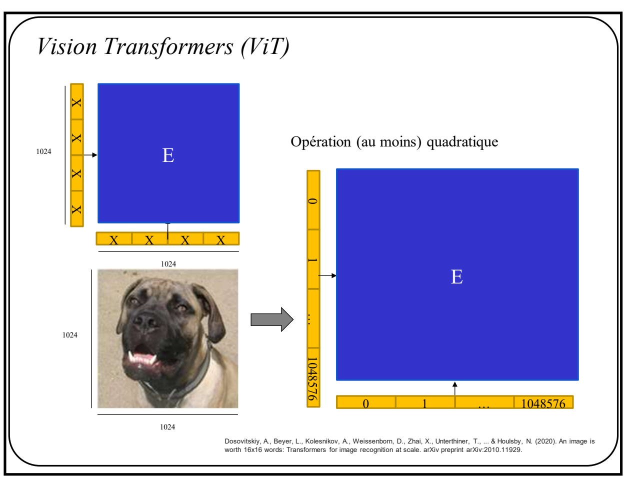
266


Vision Transformers (ViT)
Opération (au moins) quadratique
E
X X X X
1024
1024
1024
1024
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.

267

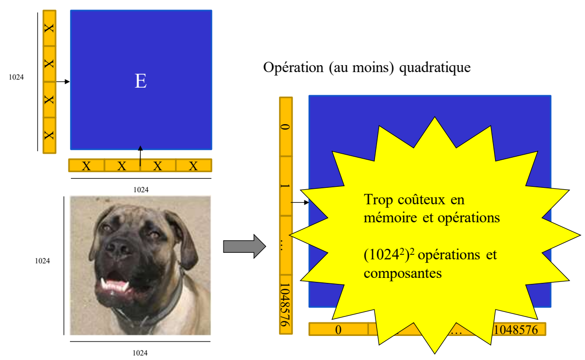

Vision Transformers (ViT)
Opération (au moins) quadratique
E
X X X X
1024
1024
1024
1024
0 1 ... 1048576
0 1 ... 1048576
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

268


Vision Transformers (ViT)
Opération (au moins) quadratique
E
X X X X
1024
1024
1024
1024
E
0 1 ... 1048576
0 1 ... 1048576
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.

269

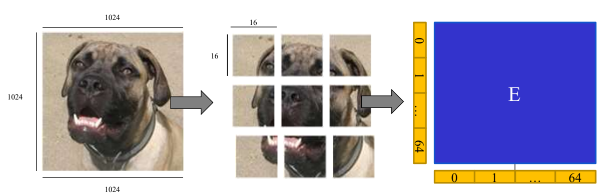
Vision Transformers (ViT)



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

270

Vision Transformers (ViT)



Chaque "patch" est un token de (par exemple) 256 composantes

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)

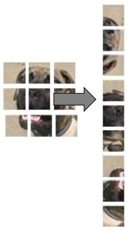
- L'image est séparée en patch

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

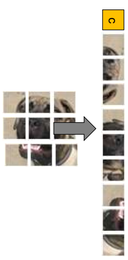


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

273

Vision Transformers (ViT)

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

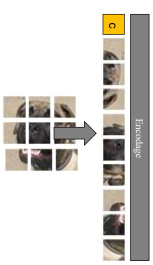


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

274

Vision Transformers (ViT)

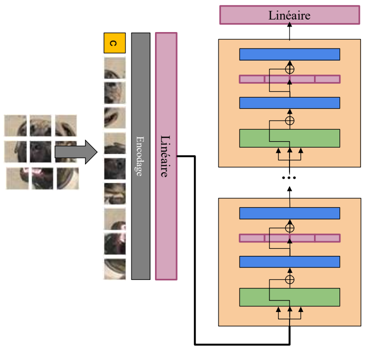
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens



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.

275

Vision Transformers (ViT)

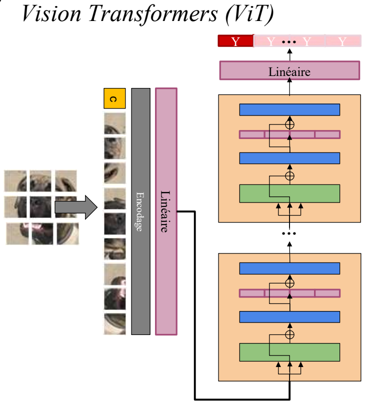


- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all you need*

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.

276

Vision Transformers (ViT)



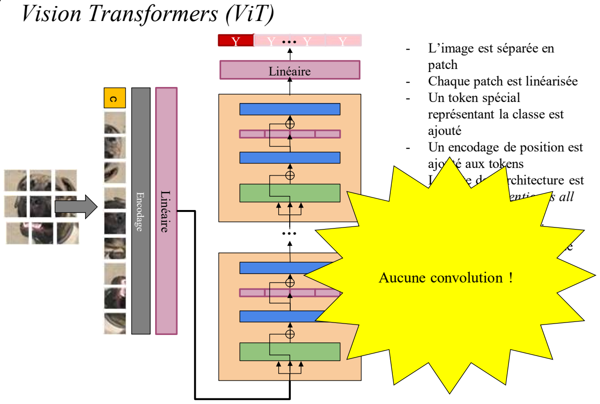
- L'image est séparée en patch
- Chaque patch est linéarisée
- Un token spécial représentant la classe est ajouté
- Un encodage de position est ajouté aux tokens
- Le reste de l'architecture est identique à *Attention is all you need*
- Seulement la prédiction correspondant au token de classe est faite

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)

- 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.

278

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** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 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.

279

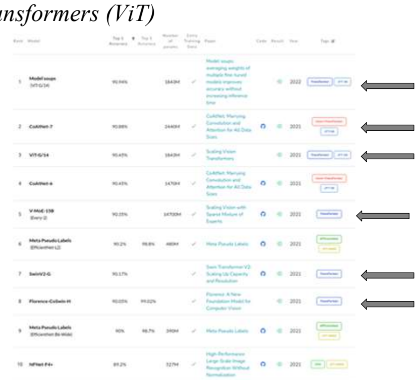
Vision Transformers (ViT)

Les *vision transformers* dominent la classification depuis leur arrivée

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

280

Vision Transformers (ViT)

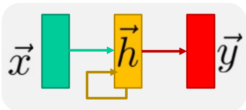


Les *vision transformers* dominent la classification depuis leur arrivée

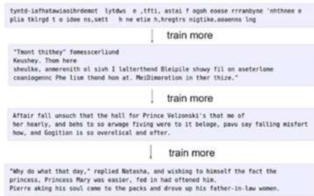
https://paperswithcode.com/sota/image-classification-on-imagenet

281

Sommaire



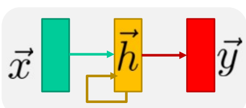
- Les réseaux récurrents peuvent traiter des séquences
- Ils ne requièrent que de légères modifications à des réseaux pleinement connectés
- Ils sont instables sur de longues séquences
- LSTM/GRU sont utilisés en pratique



Modélisation de langage

282

Sommaire



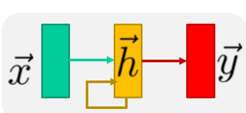
- Les réseaux récurrents peuvent traiter des séquences
- Ils ne requièrent que de légères modifications à des réseaux pleinement connectés
- Ils sont instables sur de longues séquences
- LSTM/GRU sont utilisés en pratique

a group of people playing a game with nintendo wii controllers

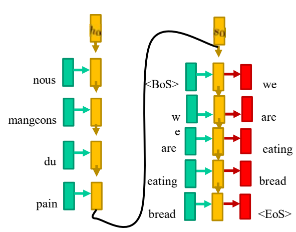
Description d'images

283

Sommaire



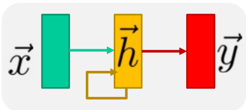
- Les réseaux récurrents peuvent traiter des séquences
- Ils ne requièrent que de légères modifications à des réseaux pleinement connectés
- Ils sont instables sur de longues séquences
- LSTM/GRU sont utilisés en pratique



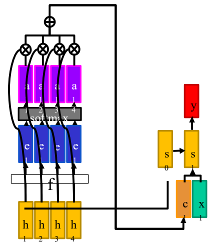
Traduction

284

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

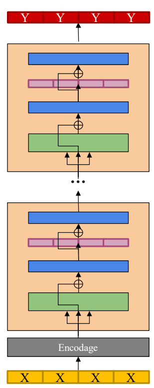


- L'attention est un mécanisme très puissant permettant aux réseaux d'apprendre quelle partie des données utilisées pour faire une prédiction
- L'attention n'est pas limitée au texte, ou même aux séquences

285

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



286