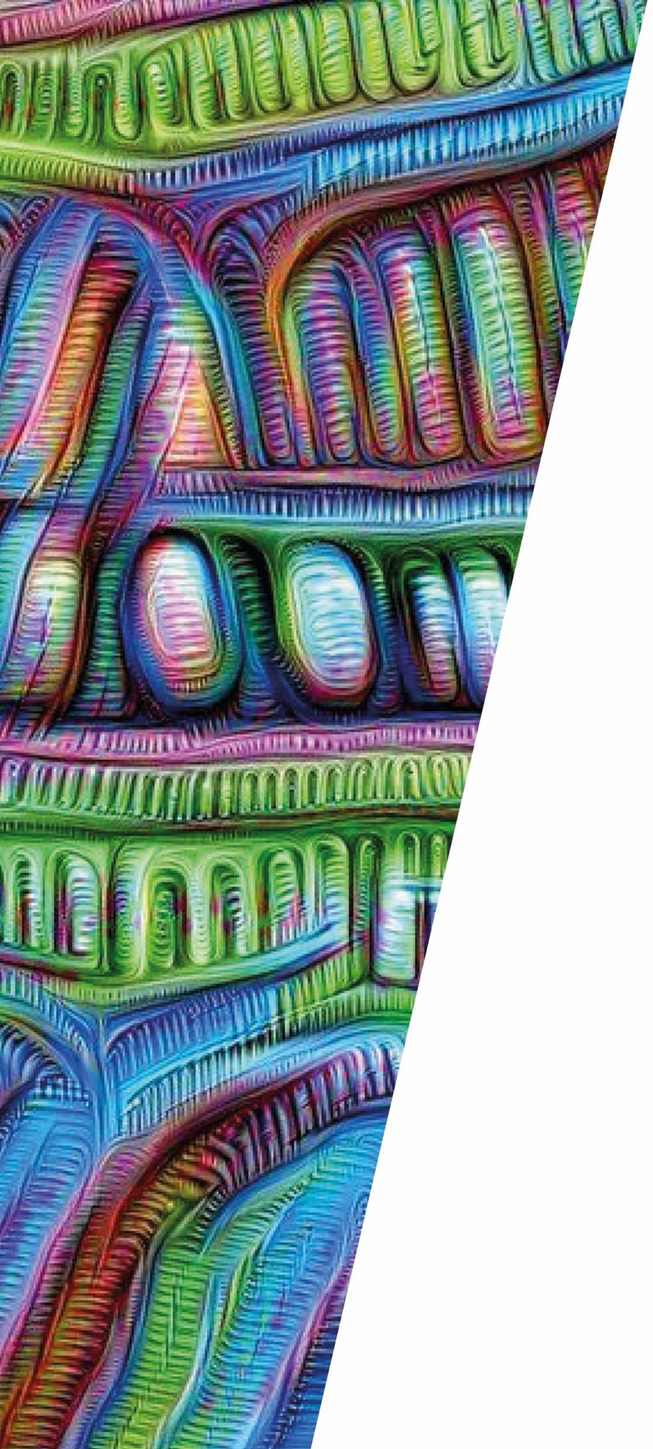


Bienvenue!

ÉCOLE D'ÉTÉ FRANCOPHONE EN APPRENTISSAGE PROFOND

21-25 août 2017

IVADO

HEC Montréal
Polytechnique Montréal
Université de Montréal

MILA

Institut
des algorithmes
d’apprentissage
de Montréal

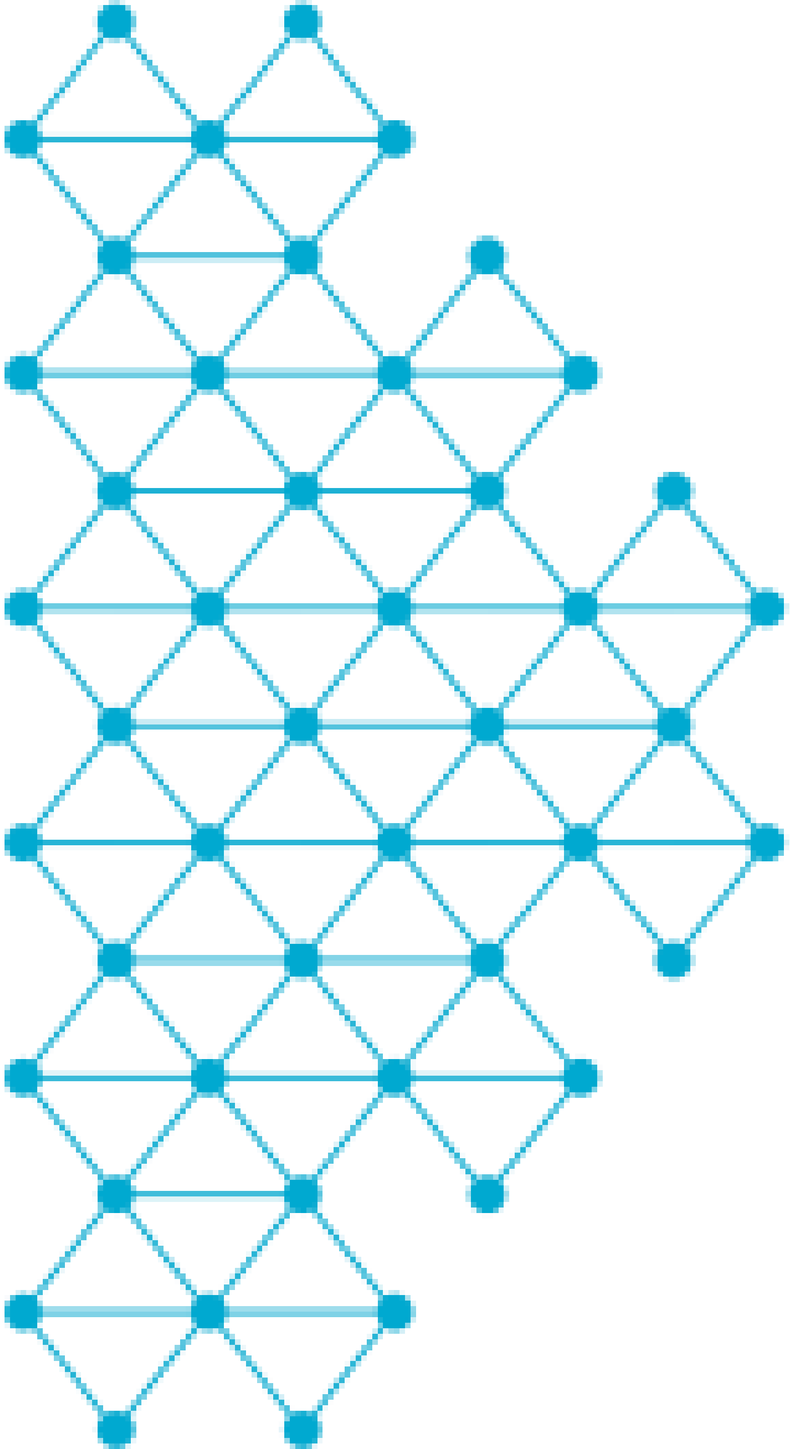
Réseaux Récurrents II

Ecole d’été IVADO

César Laurent

Plan

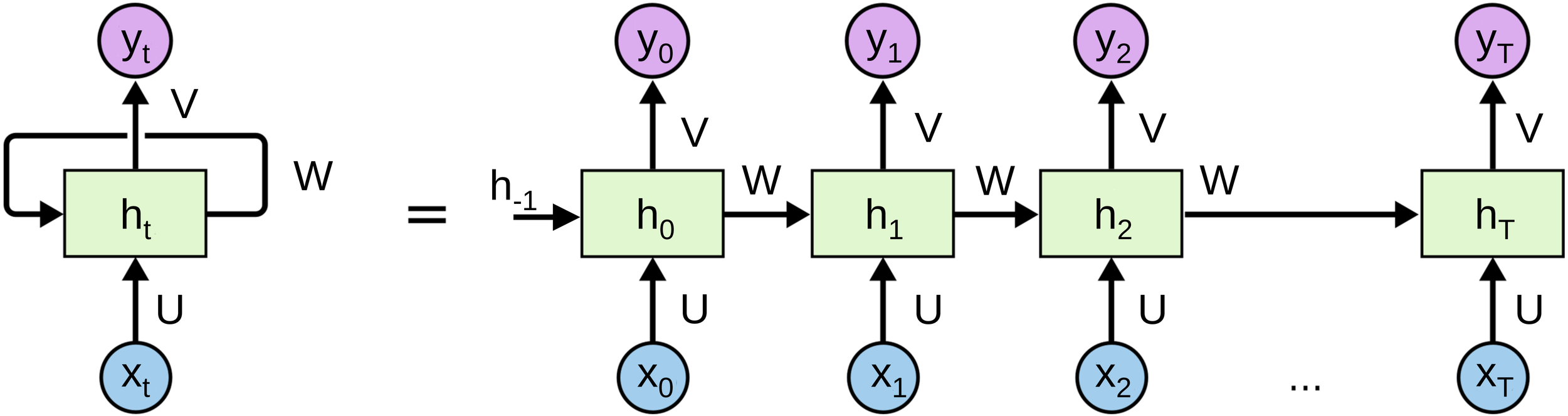
1. **Retour sur la première partie**
2. **RNNs profonds**
3. **Mécanismes d’attention**
4. **Librairies et références**



1. Retour sur la première partie

Réseaux Récurrents

Exemple déroulé dans le temps



Les paramètres sont **partagés** à travers le temps!

Image from Christopher Olah's blog

Rétropropagation à travers le temps

Exemple: Calcul du gradient sur U

- Toutes les contributions sont sommées pour obtenir le gradient sur U:

$$\frac{\partial E}{\partial U} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial U}$$

- Les gradients sur les autres paramètres sont calculés de la même manière.

$$\frac{\partial E}{\partial V} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial V} \qquad \frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W}$$

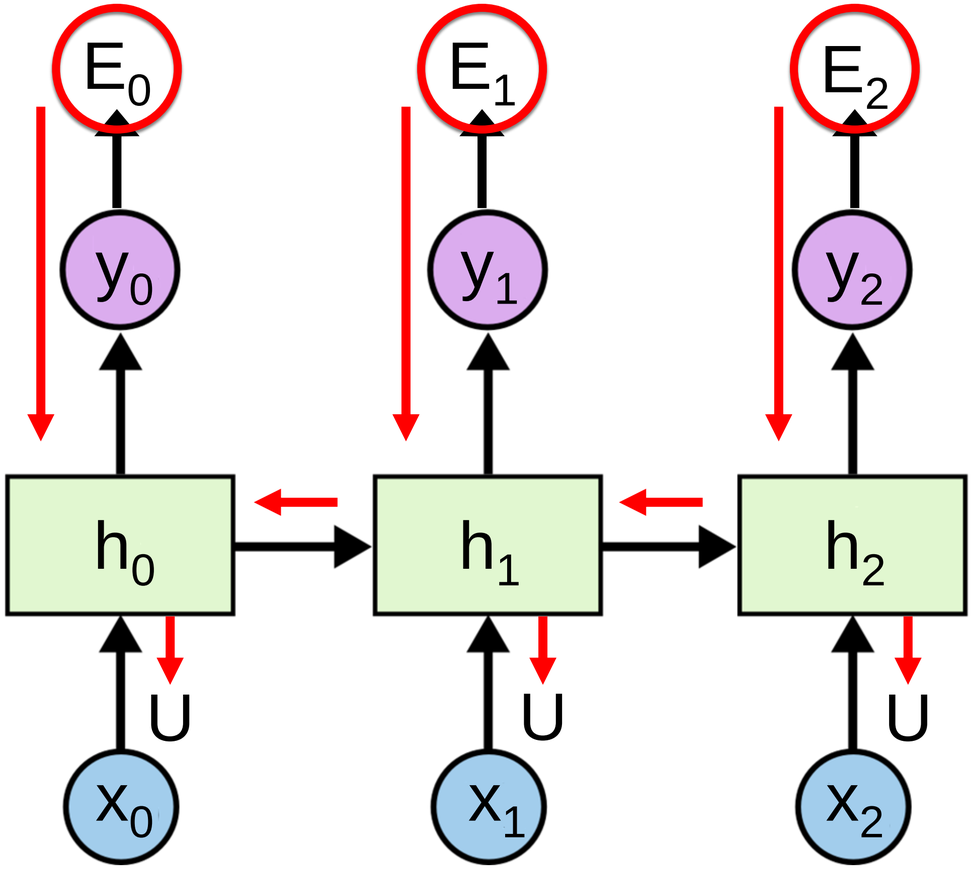
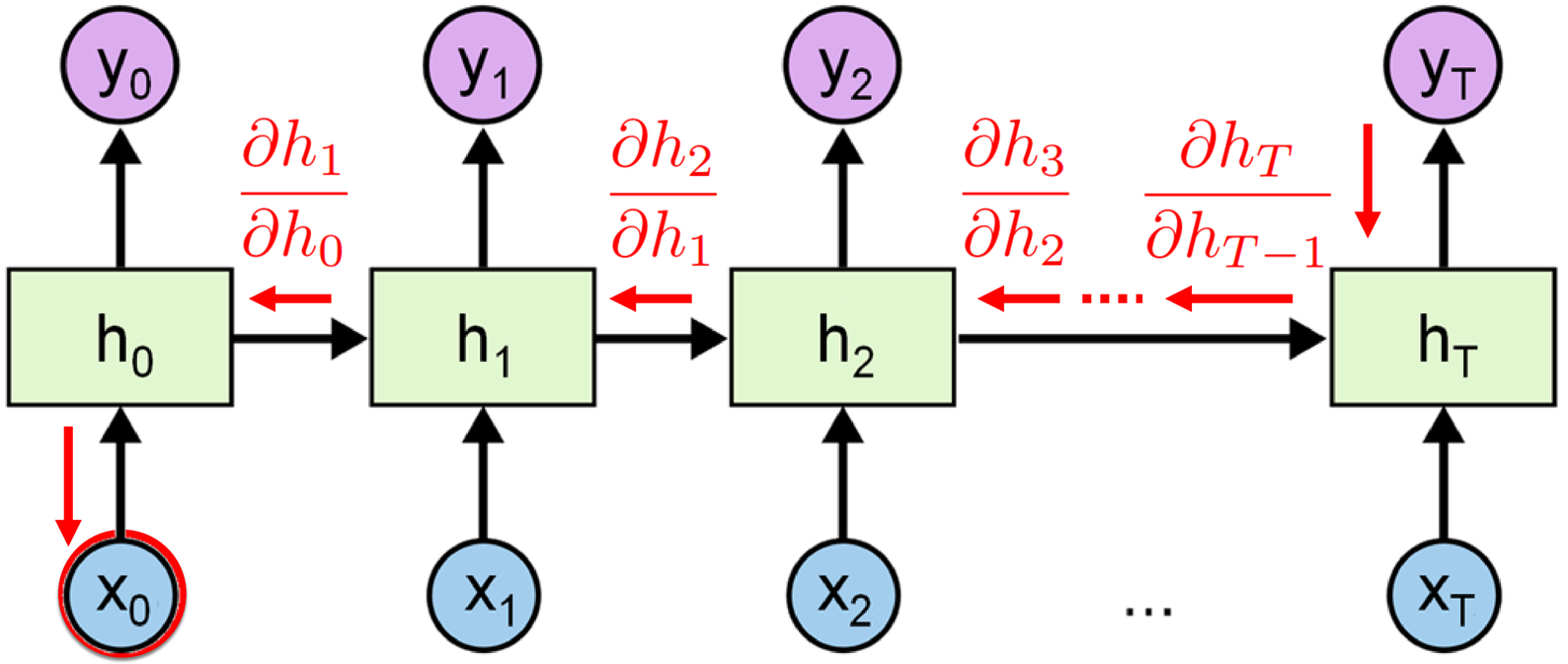


Image from Christopher Olah's blog

Dépendances à long terme

Propagation du gradient



Apprendre des dépendances à long terme

= Propager le gradient loin dans le temps

Image from Christopher Olah's blog

Long Short-Term Memory (LSTM)

En entier

Réduction du probème de disspation avec **un méchanisme de gates** et une **cellule mémoire**.

$$
\begin{aligned}
i_t &= \sigma(U_i x_t + W_i h_{t-1} + b_i) \\
f_t &= \sigma(U_f x_t + W_f h_{t-1} + b_f) \\
o_t &= \sigma(U_o x_t + W_o h_{t-1} + b_o) \\
g_t &= \tanh(U_g x_t + W_g h_{t-1} + b_g) \\
c_t &= i_t \odot g_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

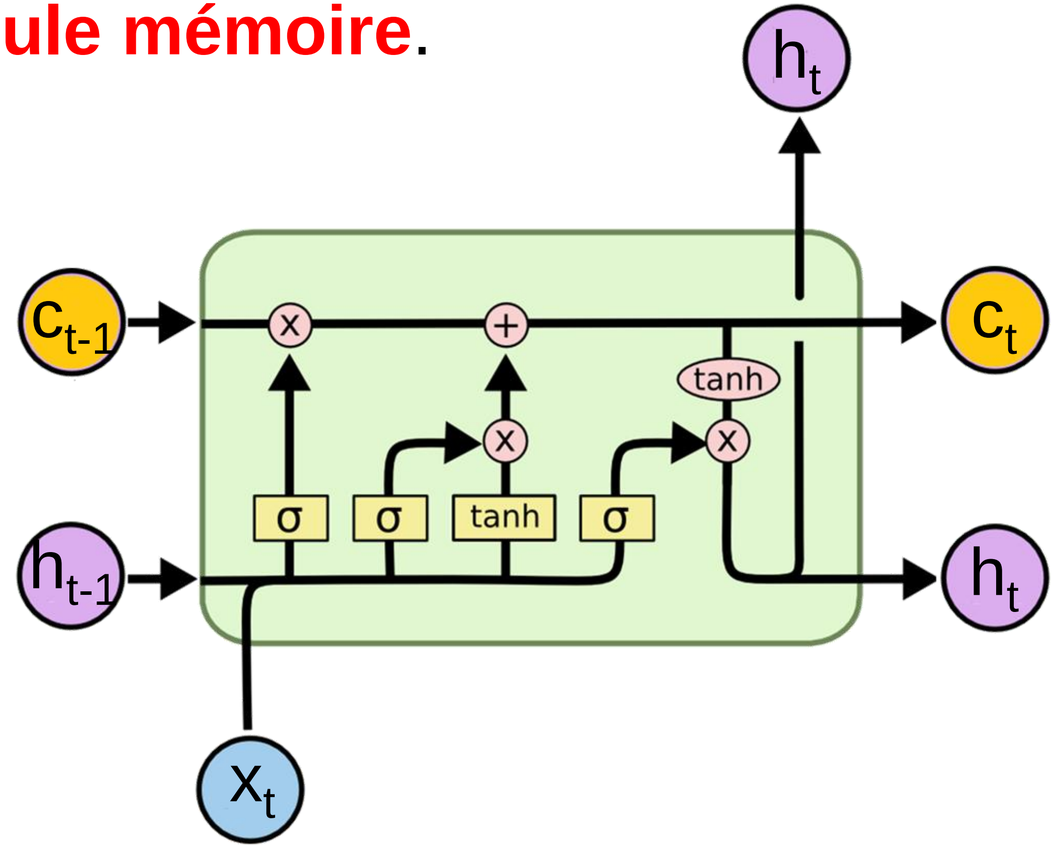


Image from Christopher Olah's blog

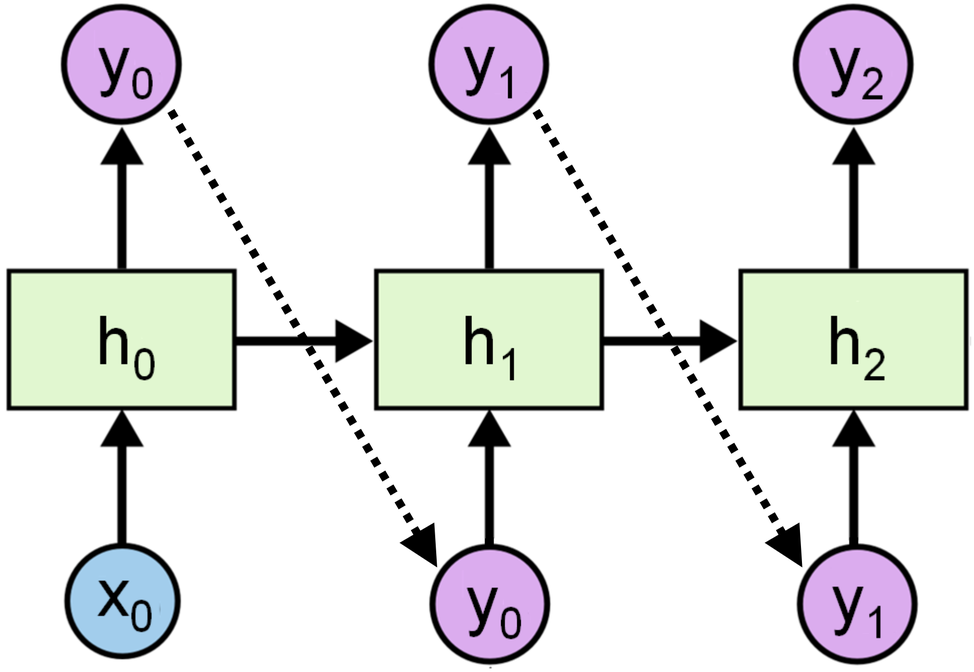
Hochreiter et al., *Long short-term memory*, Neural Computation 1997

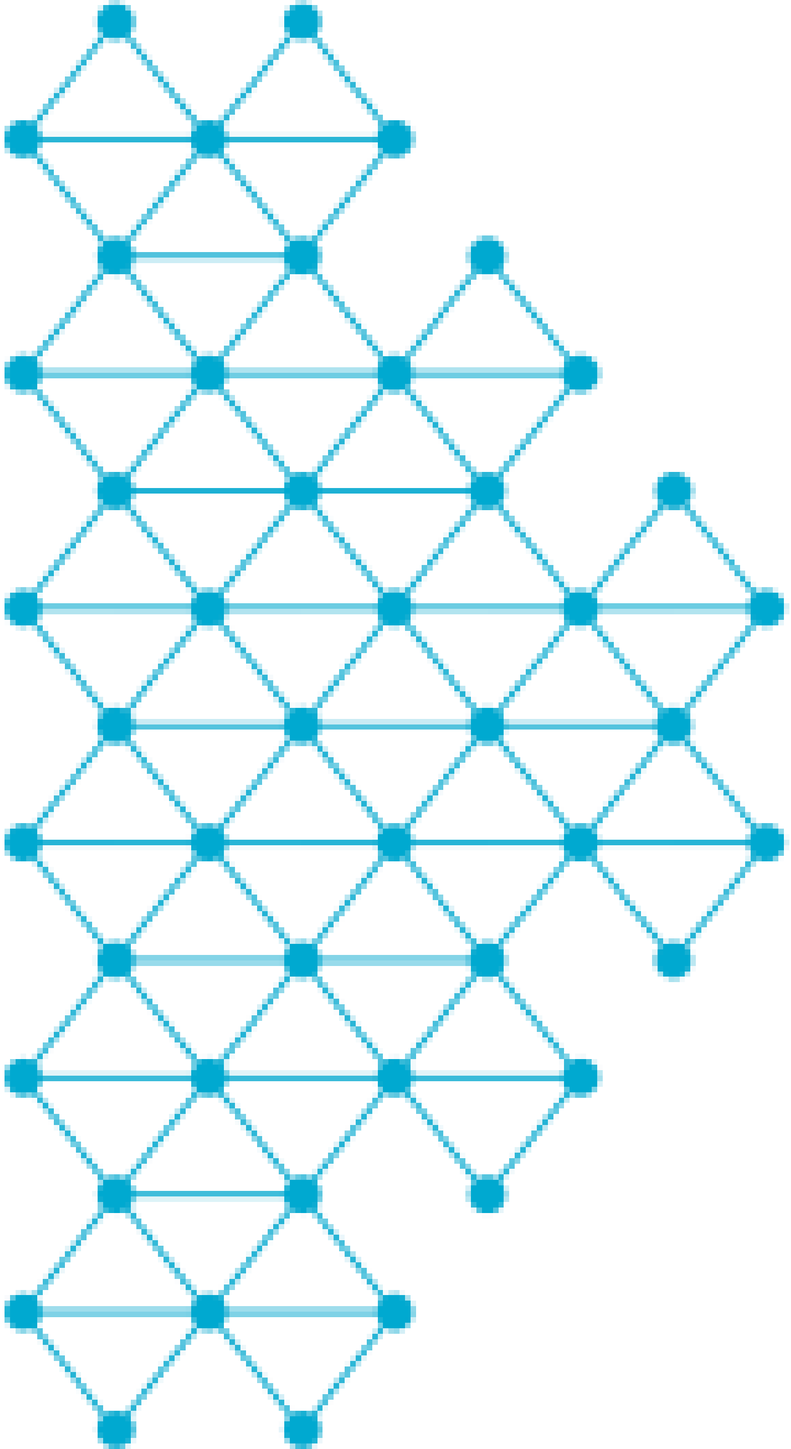
Réseaux Récurrents Génératifs

Introduction

On peut utiliser un RNN pour générer des séquences:

- On donne la sortie au temps t comme entrée au temps t+1
- Le modèle génère une séquence lui-même!

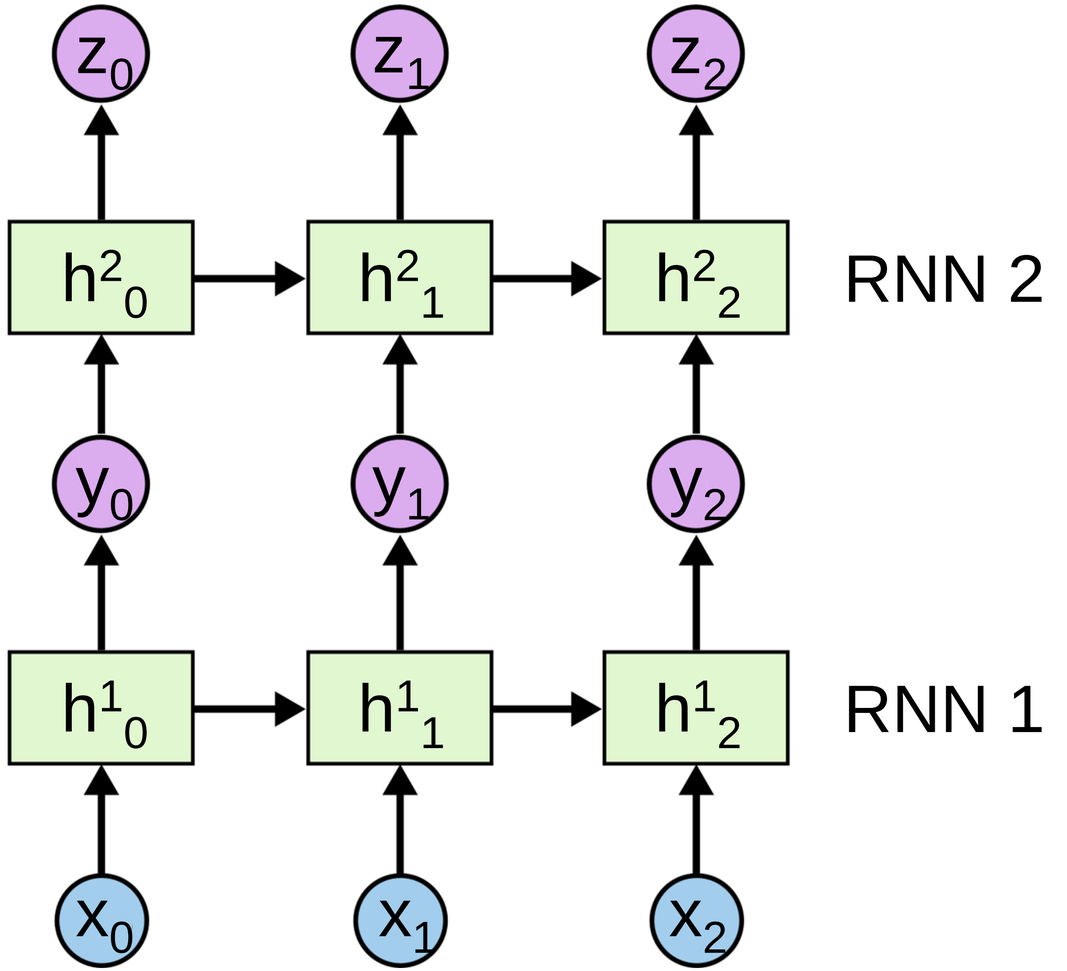




2. RNNs Profonds

Piles de Réseaux Récurrents

- Pour créer des RNNs profonds, on peut empiler des couches
  - Chaque RNN peut être un RNN simple, une LSTM, GRU,…
- La séquence de sortie de la première couche est la séquence d'entrée de la seconde couche, etc.



Réseaux Récurrents Bidirectionnels

- Ajout d'un second RNN qui lit la séquence à l'envers.
- Permet d'avoir de l'information sur ce qui se passe avant **et** après.
- Les 2 RNNs sont différents (paramètres différents)!

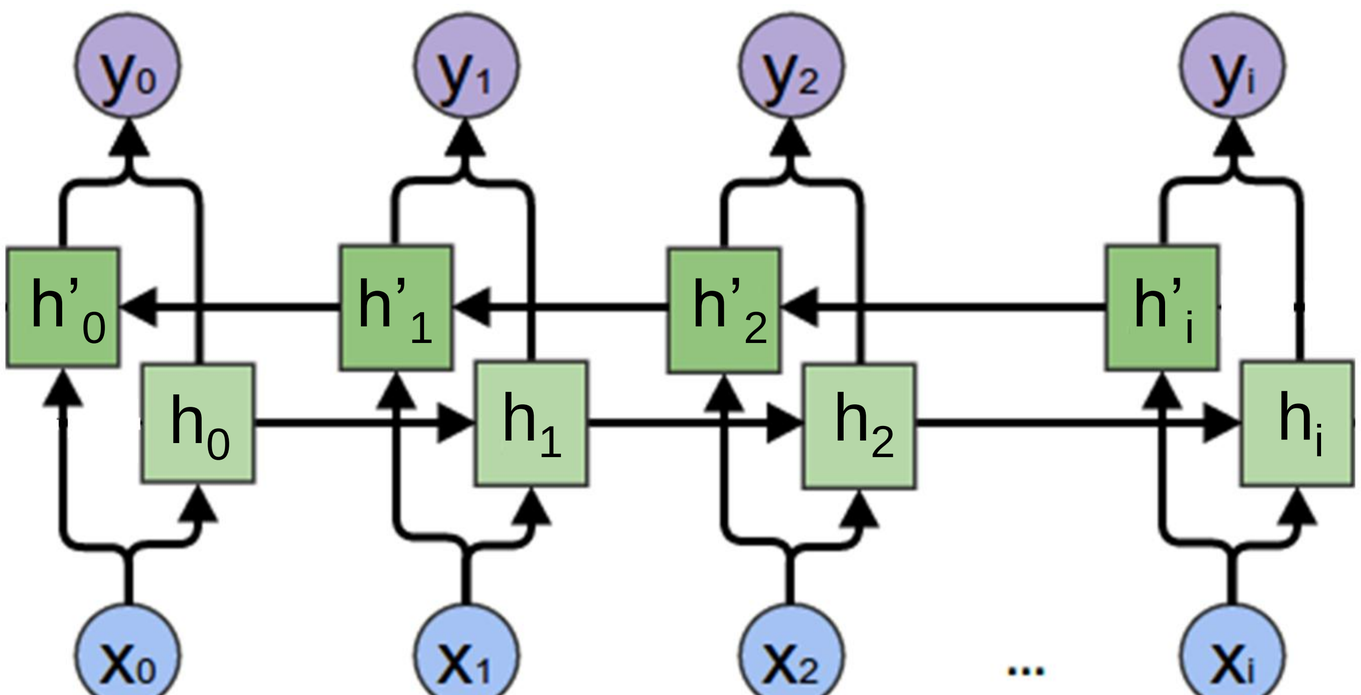


Image from Christopher Olah's blog

Détection d’événements sonores

Description de la tâche

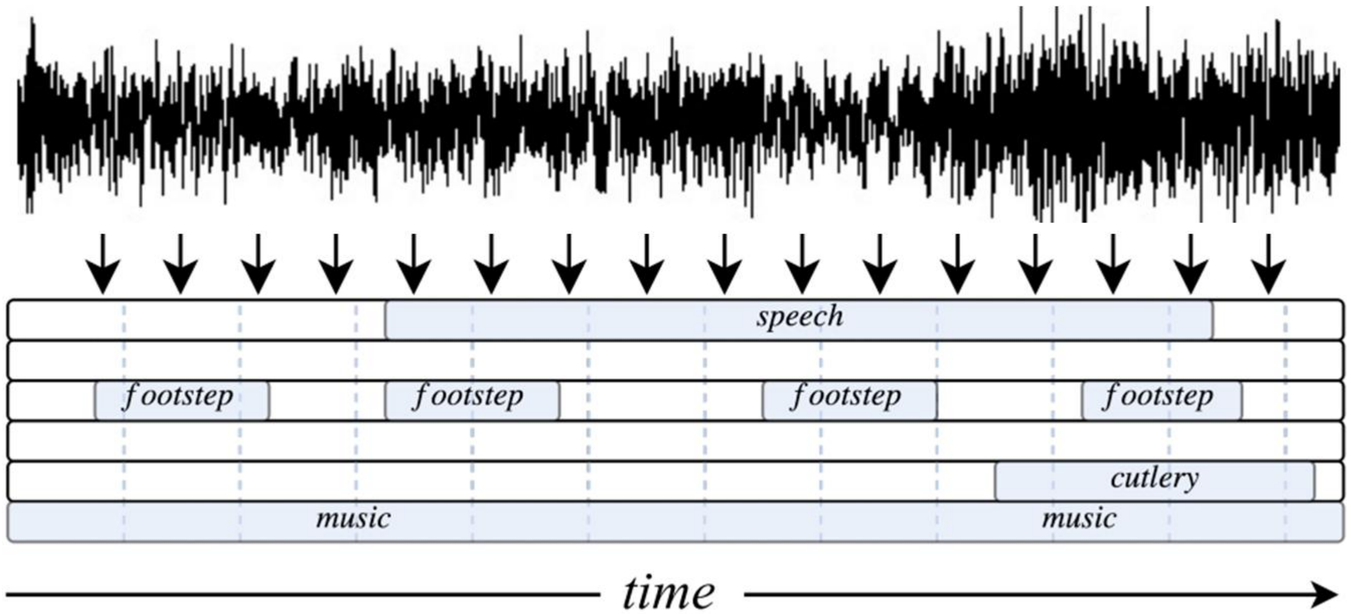
- But: Détecter et classifier des événements sonores dans des enregistrements.
  - Début et fin.
  - Type d’événement (voiture qui passe, oiseau qui chante,…).
  - Polyphonie (plusieurs événements peuvent avoir lieu en même temps).

Détection d’événements sonores

Description des données

Enregistrement brut

Cibles pour chaque classes



Données:

- 103 enregistrements (~ 20 h.)
- 10 classes d’événements (basketball, plage, bus, voiture, hall, bureau, restaurant, magasin, rue, stade) annotés (début et fin)
- Polyphonie (2.5 en moyenne)

Détection d’événements sonores

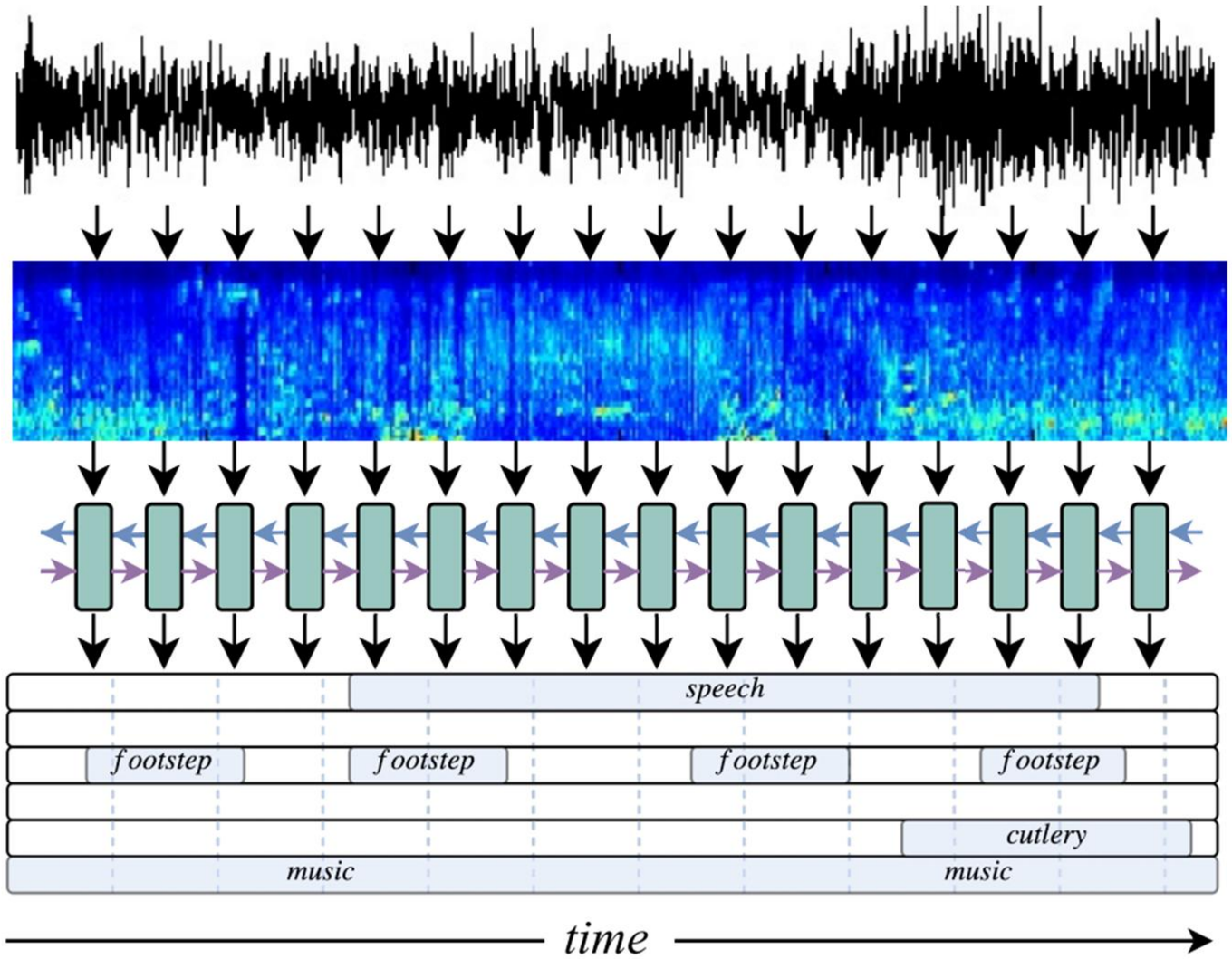
Modèle

Enregistrement brut

Spectrogramme

LSTMs bidirectionnels (4)

Cibles pour chaque classes



Détection d'événements sonores

Code

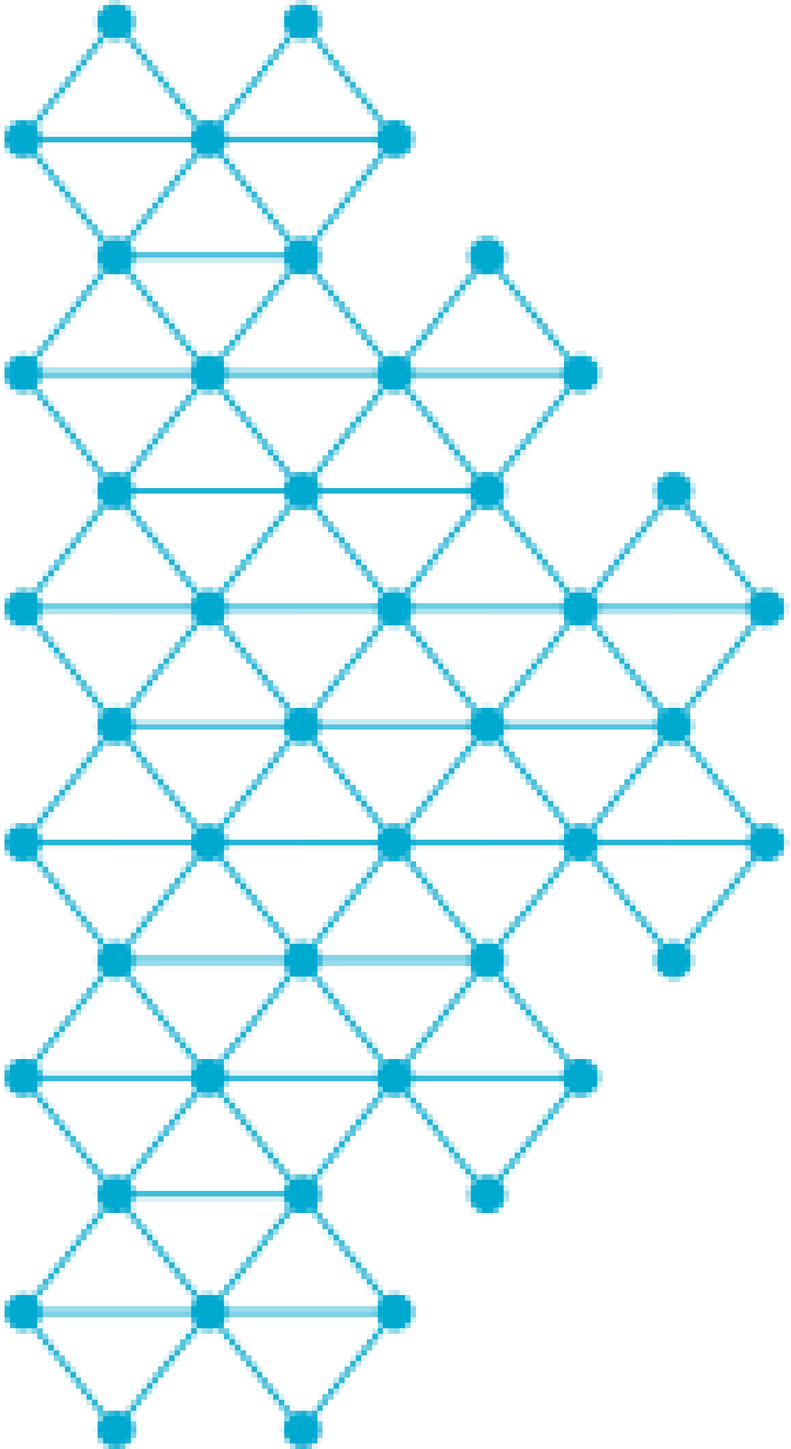
```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Merge

# First layer
fwd1 = Sequential()
fwd1.add(LSTM(100, return_sequence=True))
bwd1 = Sequential()
bwd1.add(LSTM(100, return_sequence=True, go_backwards=True))

# Second Layer
fwd2 = Sequential()
fwd2.add(LSTM(100, return_sequence=True))
bwd2 = Sequential()
bwd2.add(LSTM(100, return_sequence=True, go_backwards=True))

# Model
model = Sequential()
model.add(Merge([fwd1, bwd1], mode='sum'))
model.add(Merge([fwd2, bwd2], mode='sum'))
model.add(Dense(10), activation='sigmoid')
model.compile(loss='binary_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

Détection d'événements sonores

Résultats

Modèle	# Paramètres	F1 Score
MLP	1.65 M	63.0 %
LSTMs	850 K	63.8 %
LSTMs Bidirectionnels	850 K	**64.6 %**

- LSTMs meilleurs que MLP (même avec plus de paramètres!)
- LSTMs bidirectionnels meilleurs que LSTMs (avec même nombre de paramètres!)



3. Mécanismes d’attention

Mécanisme d’attention

Introduction

Image: https://www.banfflakelouise.com/water-activities/canoeing-kayaking

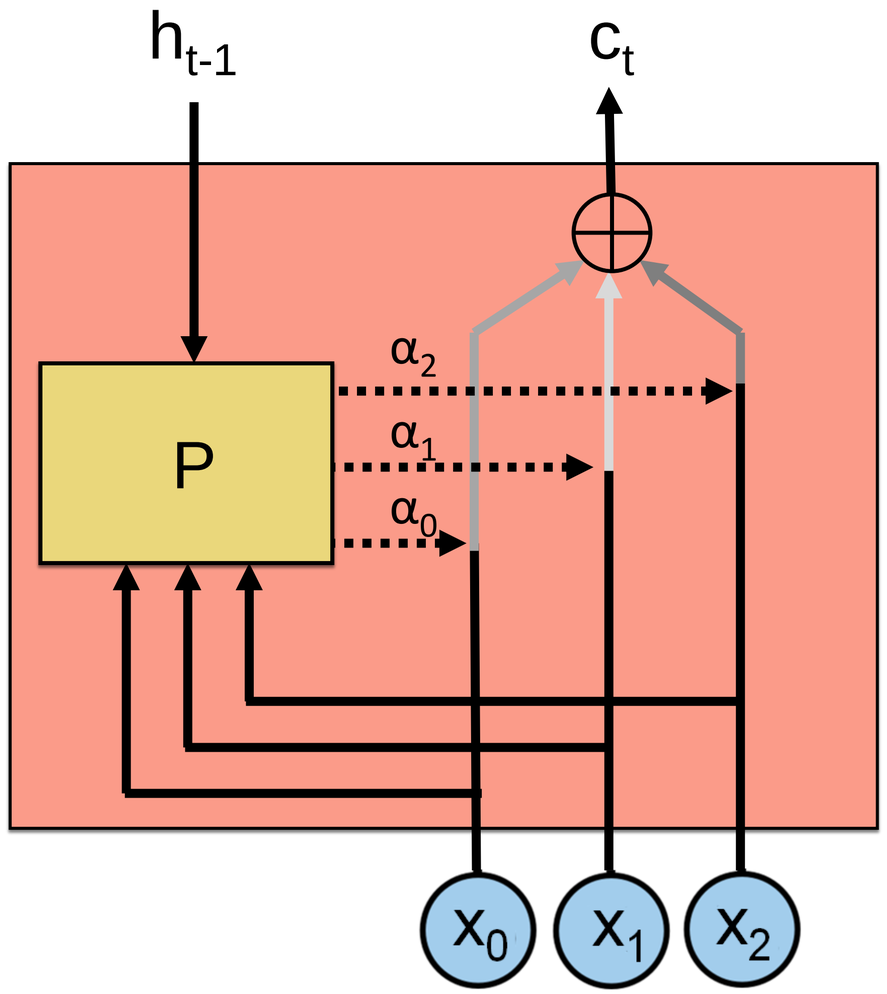
Mécanismes d'attention

Architecture

Un mécanisme d'attention permet à un RNN de porter son attention sur différentes parties de son entrée $[x_0, \ldots, x_T]$.

1. Chaque élément de l'entrée est comparé à l'état courant h_{t-1}, pour générer des coefficients de pondération α_i.

2. La sortie c_t (contexte) est une somme pondérée de tous les éléments de l'entrée (« Soft Attention »).

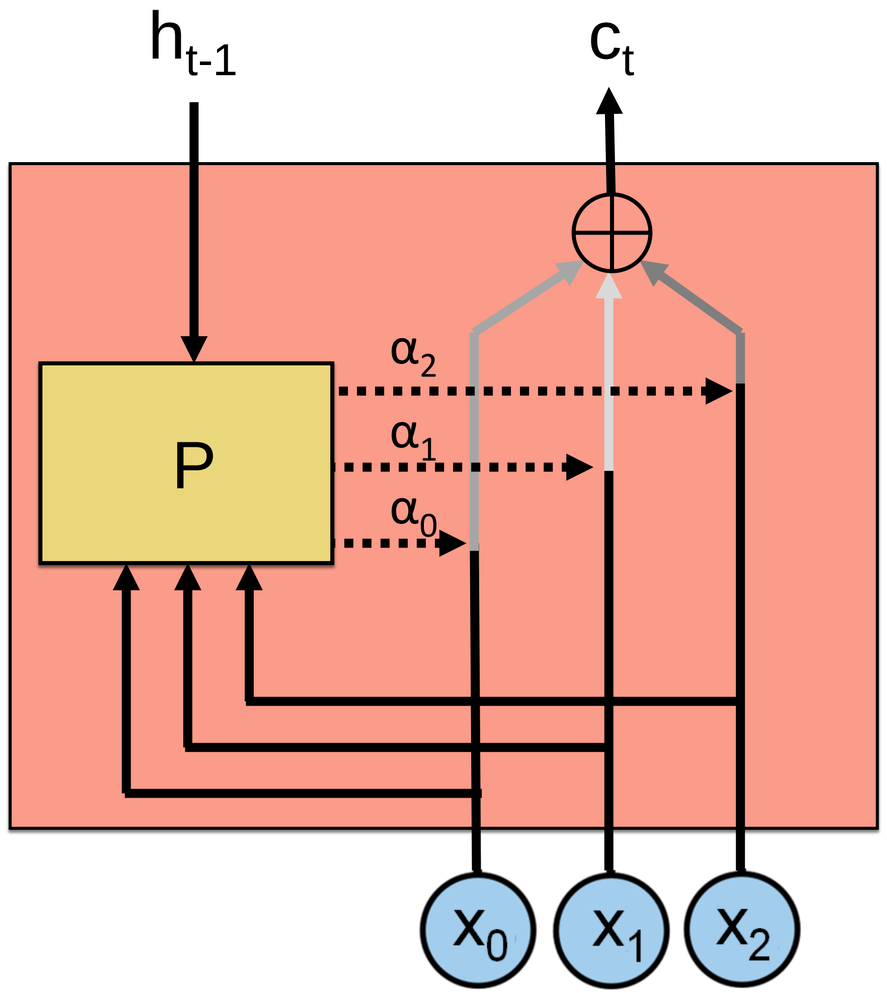


Mécanismes d’attention

Description de P

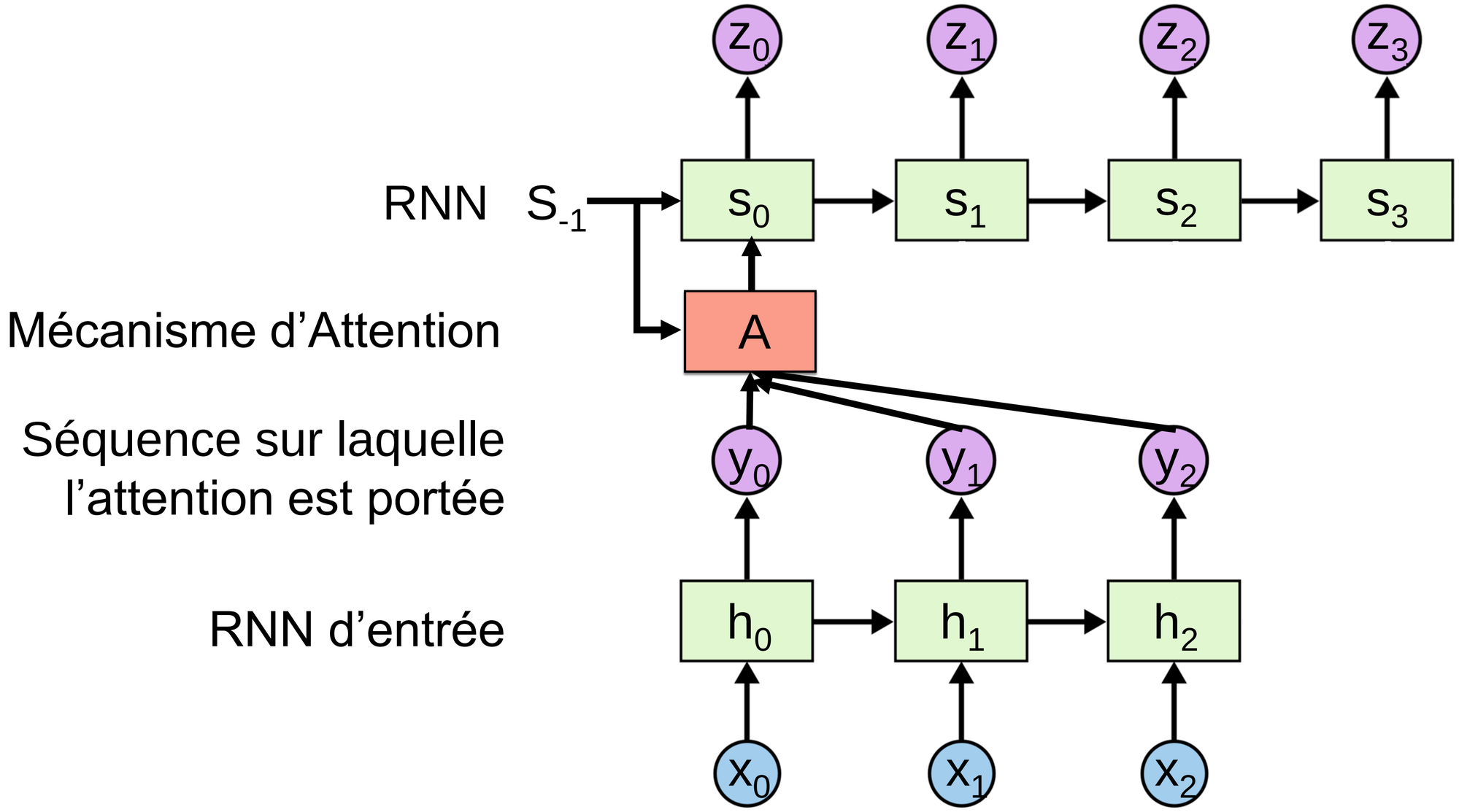
- La version la plus simple de P effectue les produits scalaires $e_i = < h_{t-1}, x_i >$ puis les normalise à l’aide d’un softmax.

- P peut aussi être un réseau de neurones (et même profond!).



Mécanismes d'attention

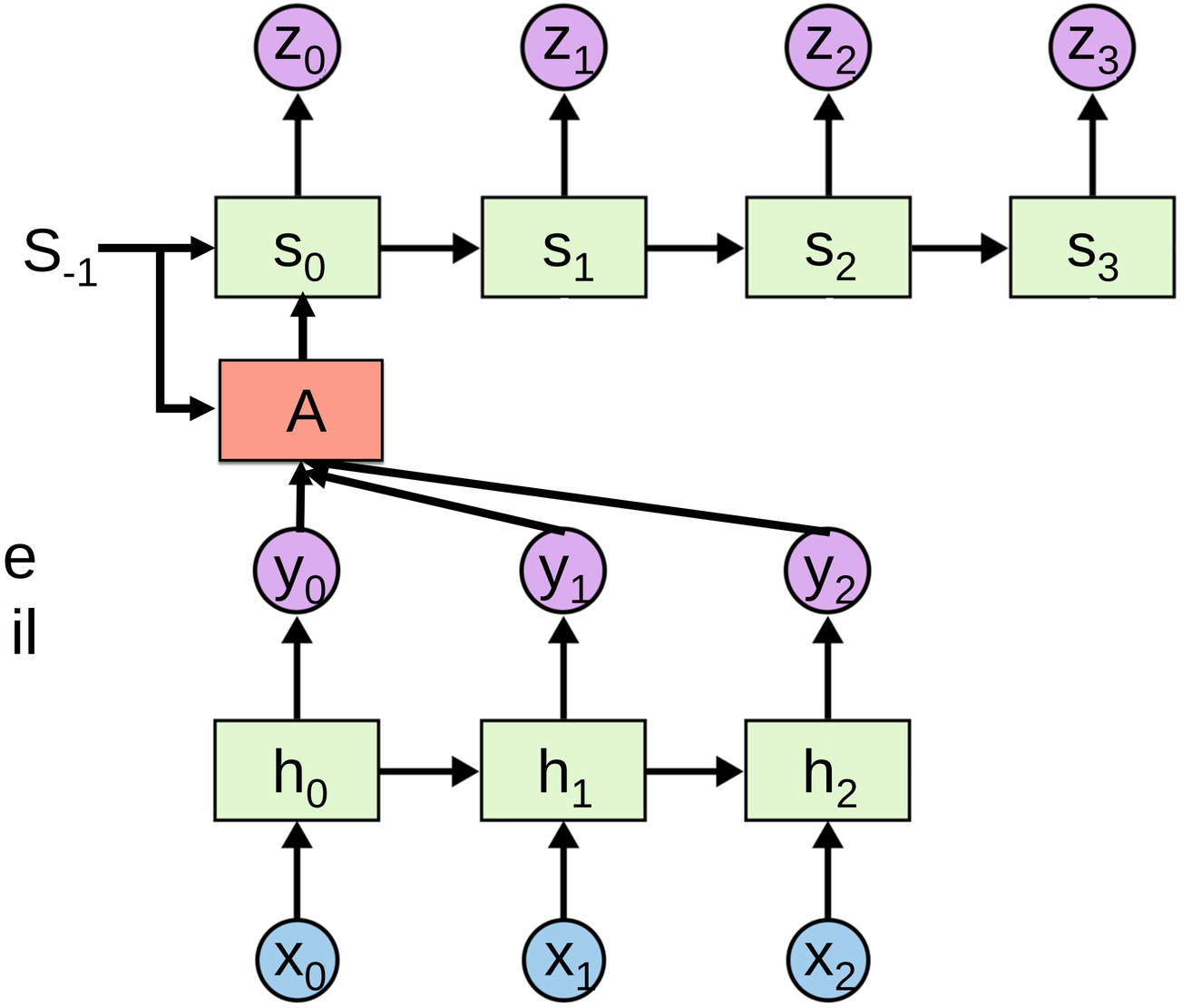
Avec RNN



Mécanismes d'attention

Premier temps

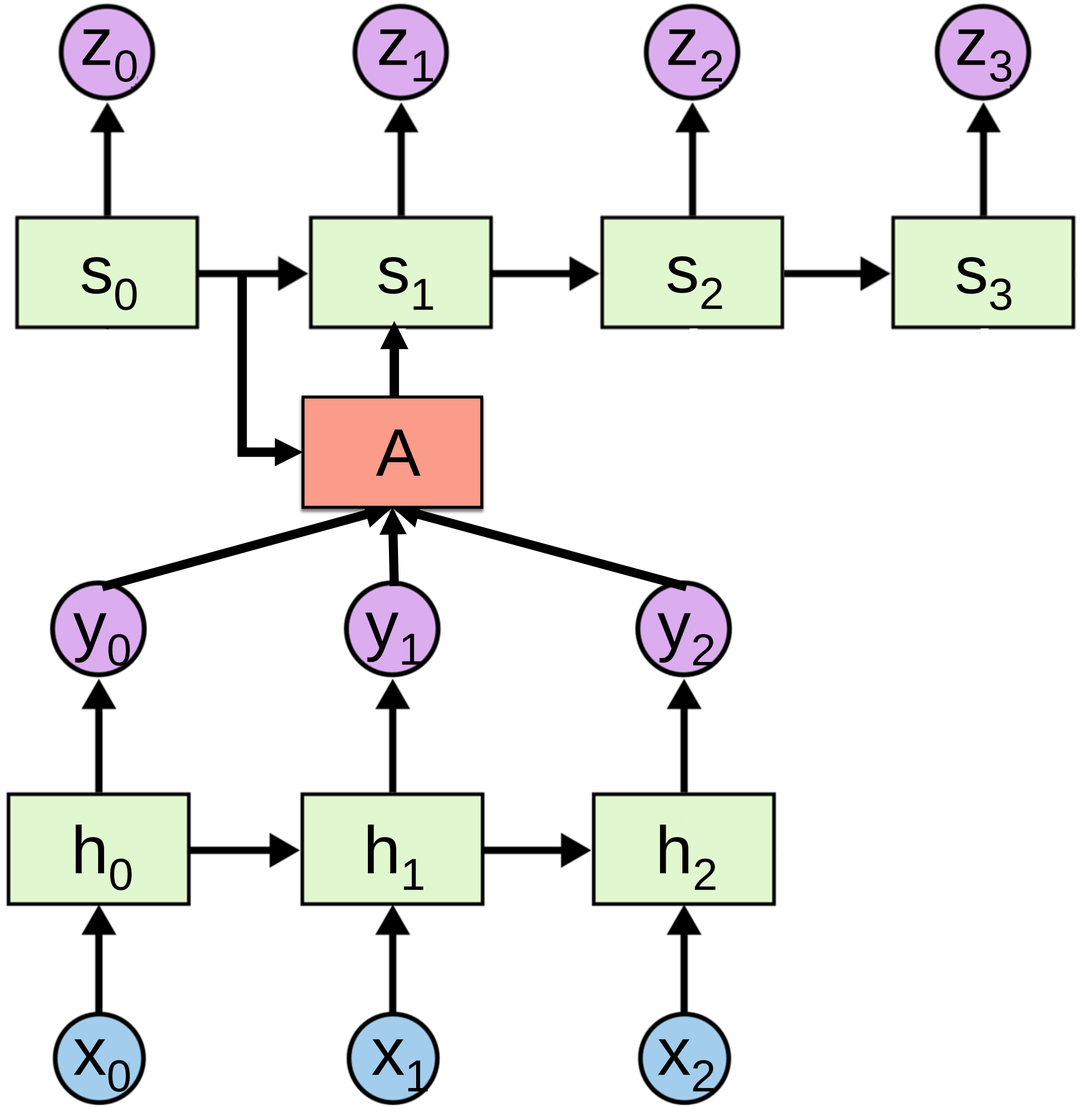
- Le mécanisme d'attention A génère l'entrée suivante du RNN à partir de l'état s_{-1} et de toute la séquence sur lequel il porte son attention.



Mécanismes d’attention

Deuxième temps

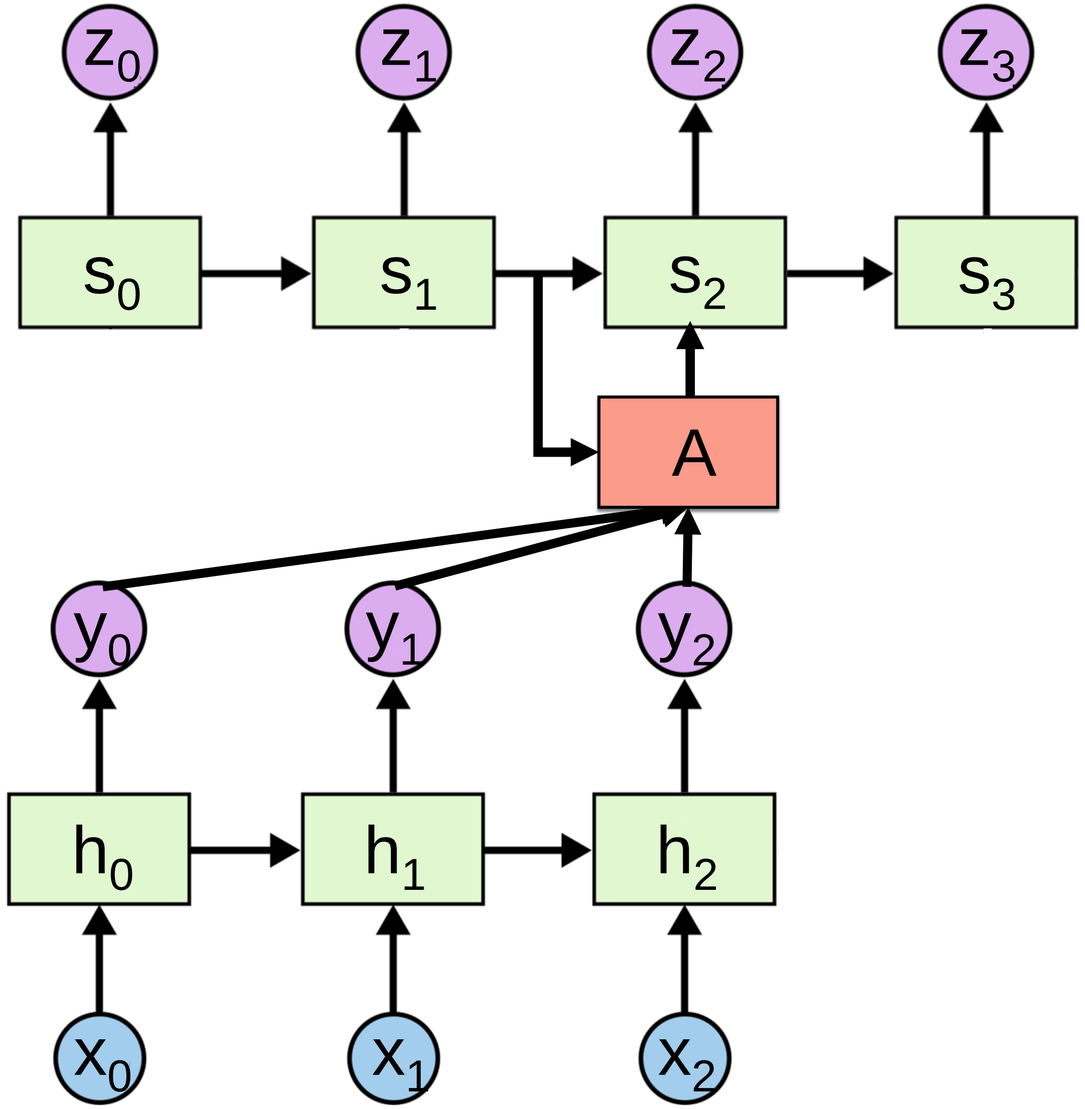
- Le mécanisme d’attention A génère l’entrée suivante du RNN à partir de l’état s_0 et de toute la séquence sur lequel il porte son attention.



Mécanismes d’attention

Troisième temps

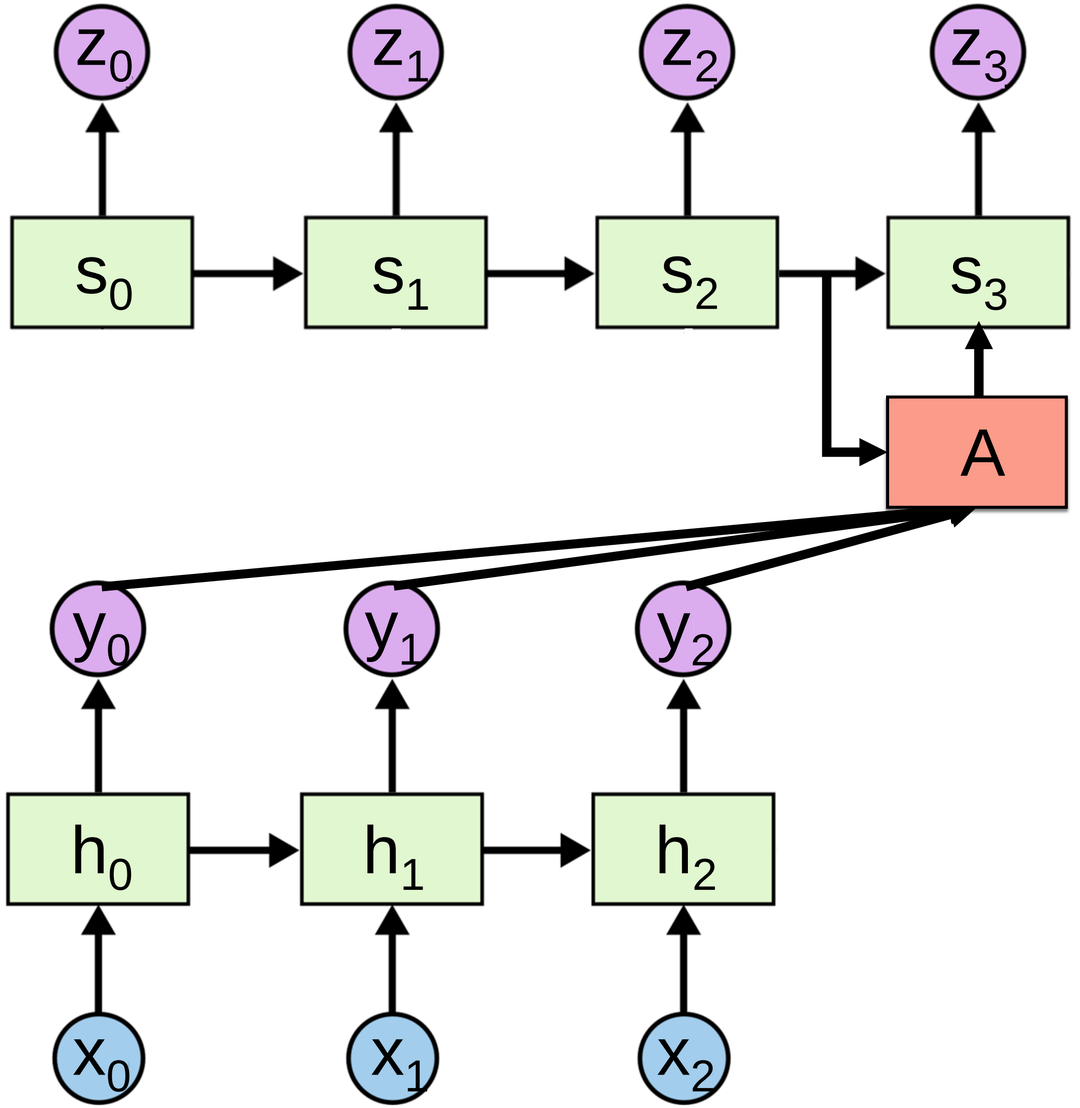
- Le mécanisme d’attention A génère l’entrée suivante du RNN à partir de l’état s_1 et de toute la séquence sur lequel il porte son attention.



Mécanismes d’attention

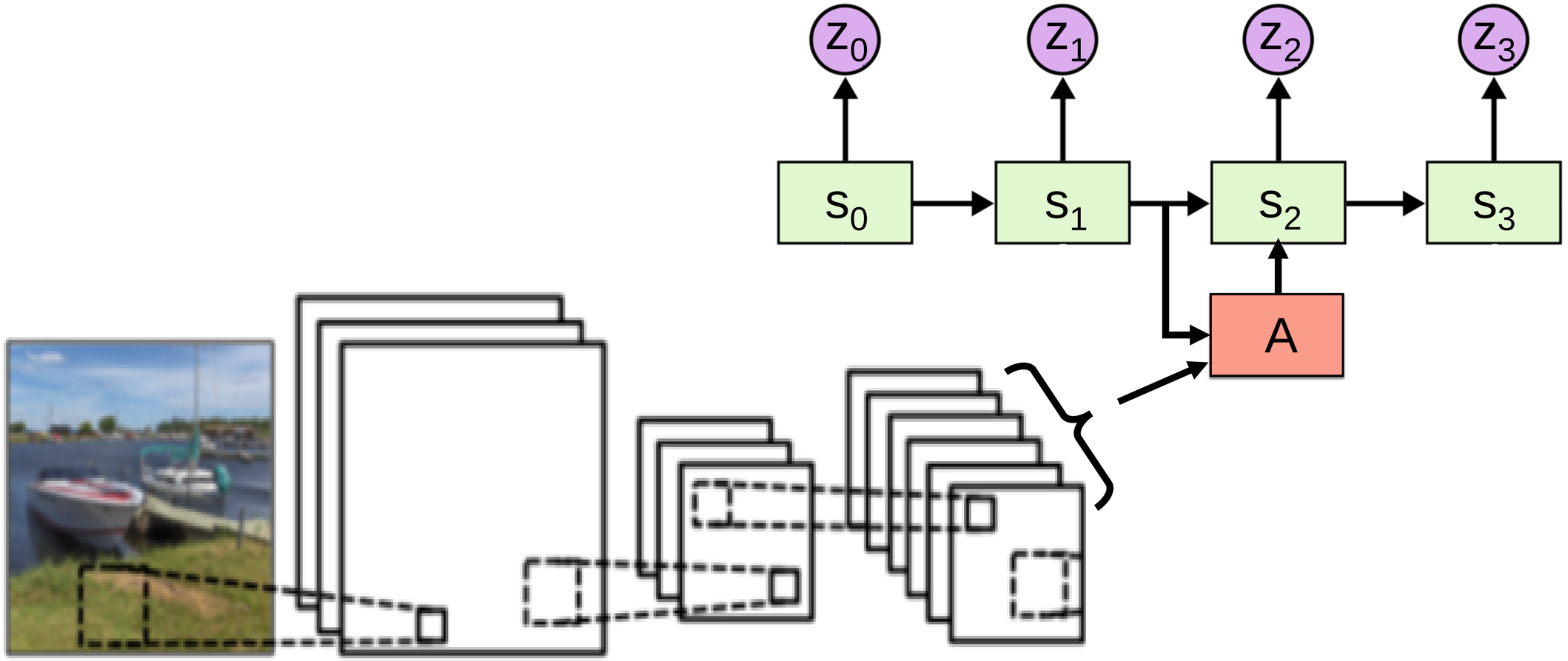
Quatrième temps

- Le mécanisme d’attention A génère l’entrée suivante du RNN à partir de l’état s_2 et de toute la séquence sur lequel il porte son attention.



Mécanismes d’attention

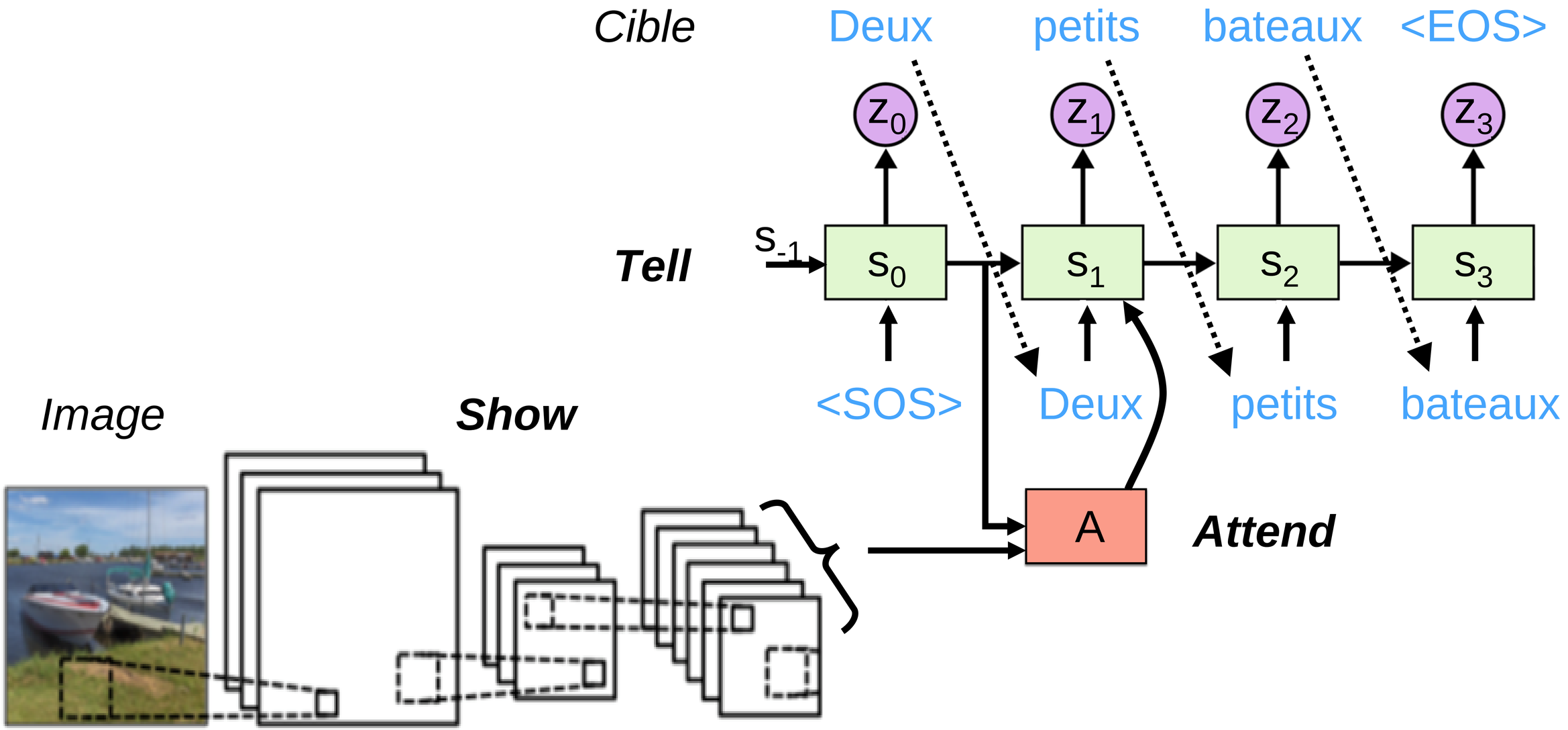
Sur une image



Un mécanisme d’attention peut aussi être utilisé sur une image!

Génération de légende

Modèle « Show Attend and Tell »



Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

En détails

Cible

Deux petits bateaux <EOS>

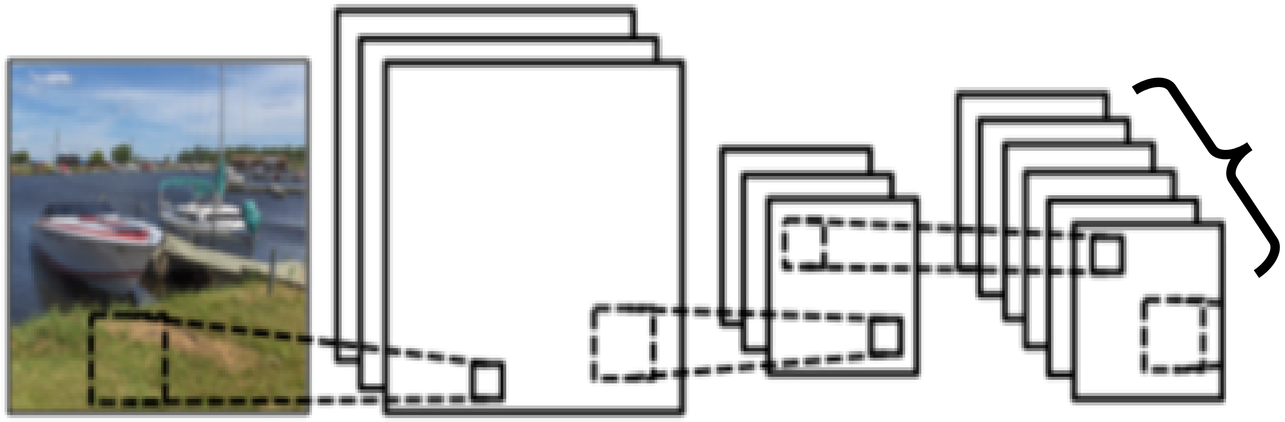
LSTM Génératif avec attention
Tâche: prédiction du prochain mot

z_0 z_1 z_2 z_3

s_{-1} s_0 s_1 s_2 s_3

<SOS> Deux petits bateaux

Image *CNN entrainé (VGG)*



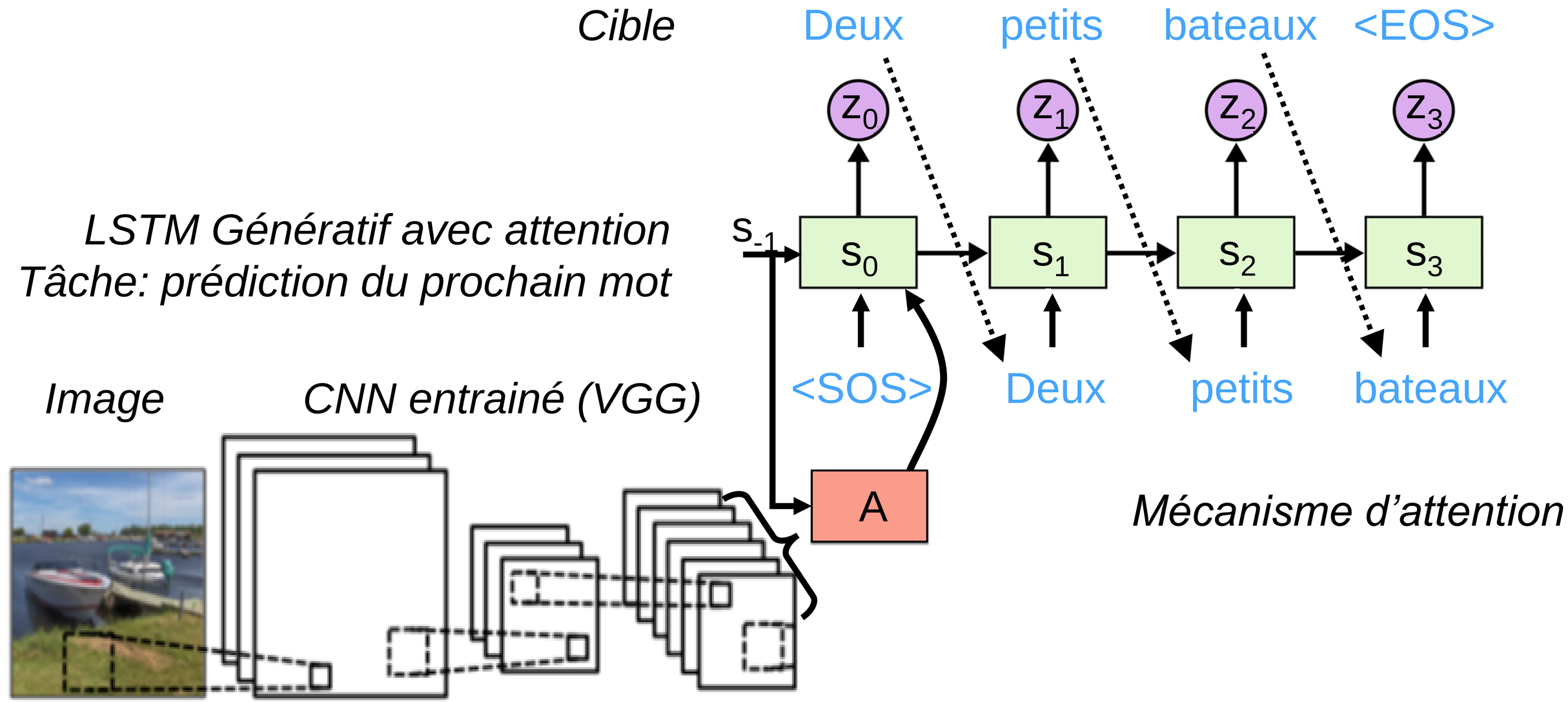
A *Mécanisme d'attention*

<SOS> Start of sequence
<EOS> End of sequence

Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

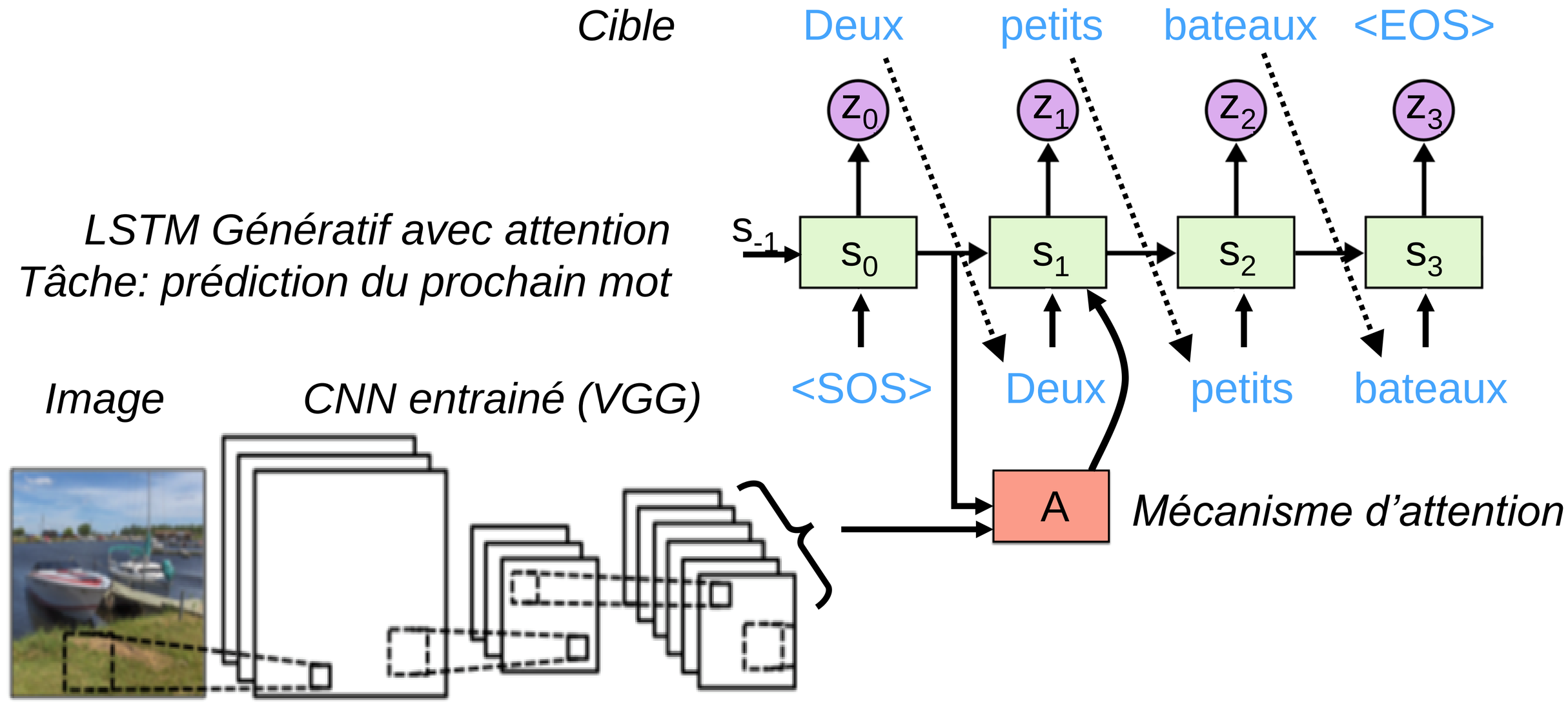
Entraînement: Génération du premier mot (a partir de <SOS>)



Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

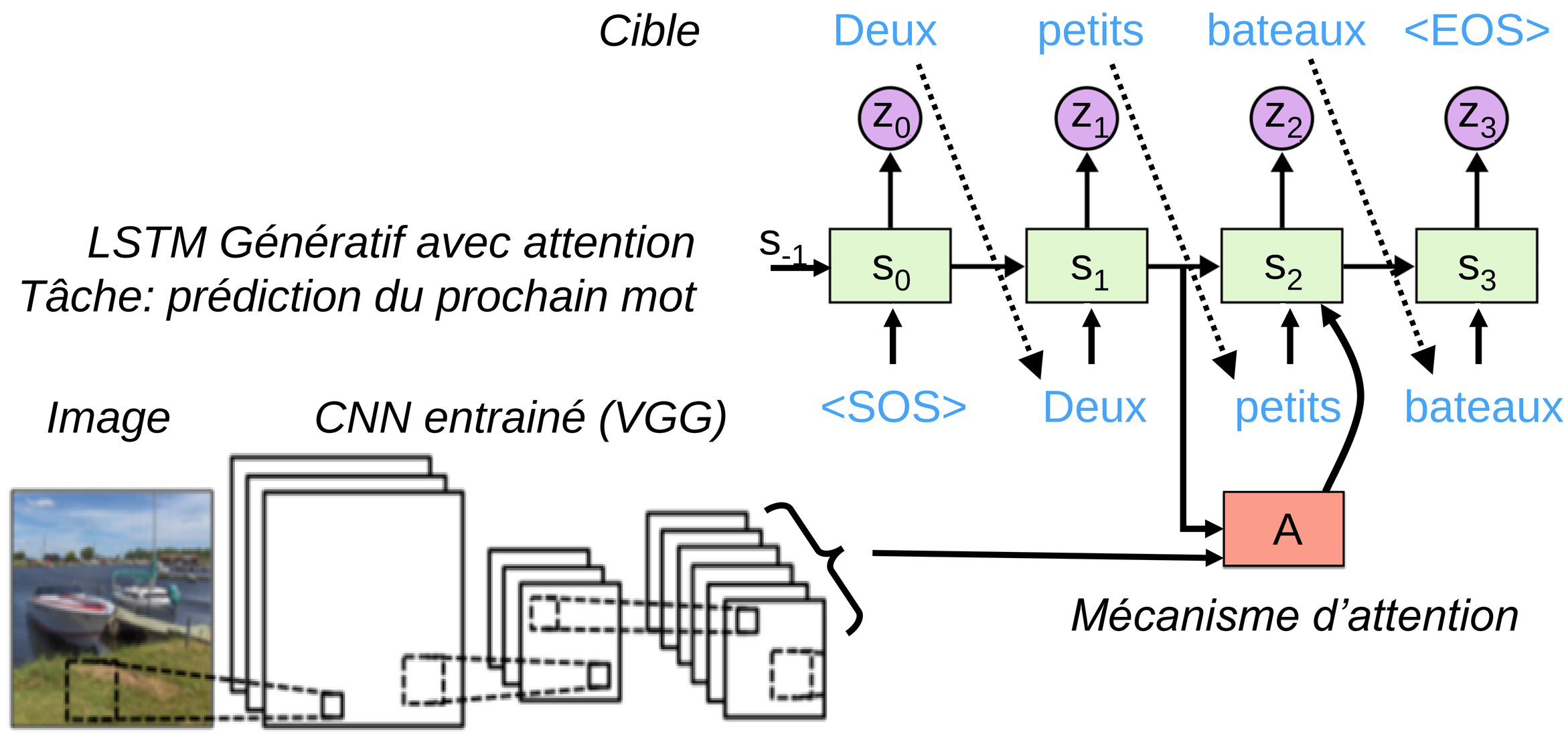
Entraînement: Génération du deuxième (a partir de « Deux »)



Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

Entraînement: Génération du troisième mot (a partir de « petits »)



Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

Test: Utilisation de la séquence z

Mot généré le plus probable: Un bateau blanc <EOS>

LSTM Génératif Avec attention

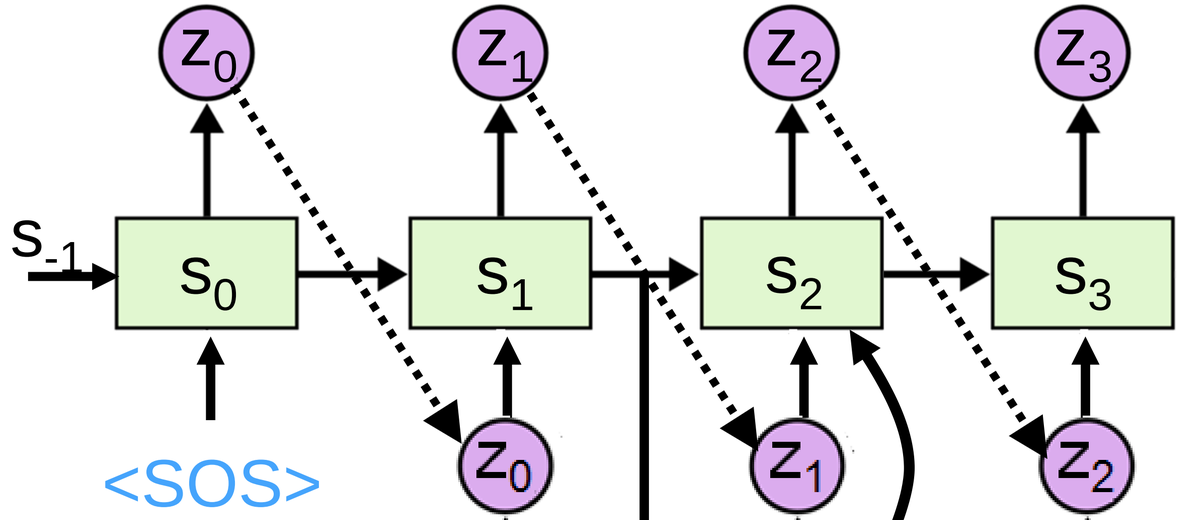
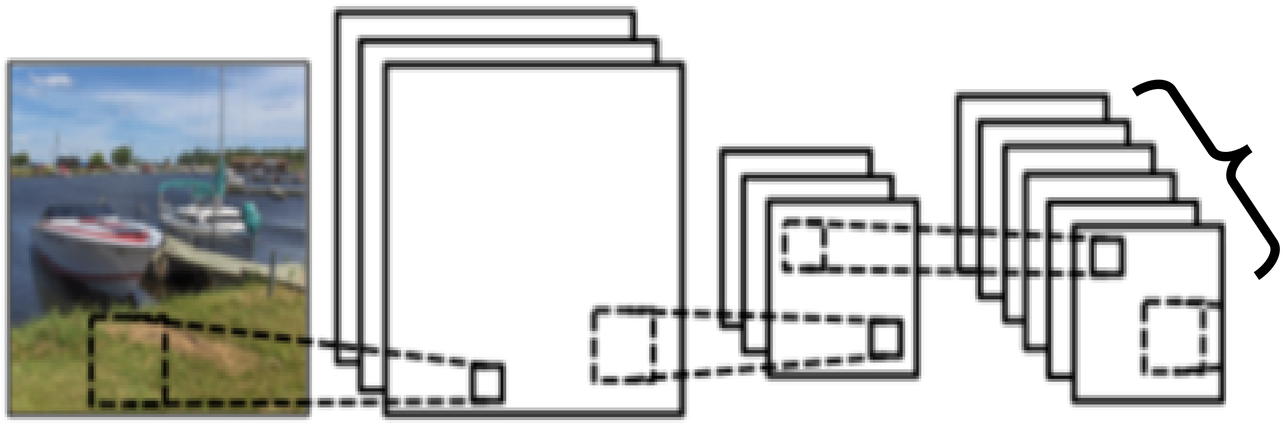


Image *CNN entrainé (VGG)*



Mécanisme d'attention

Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

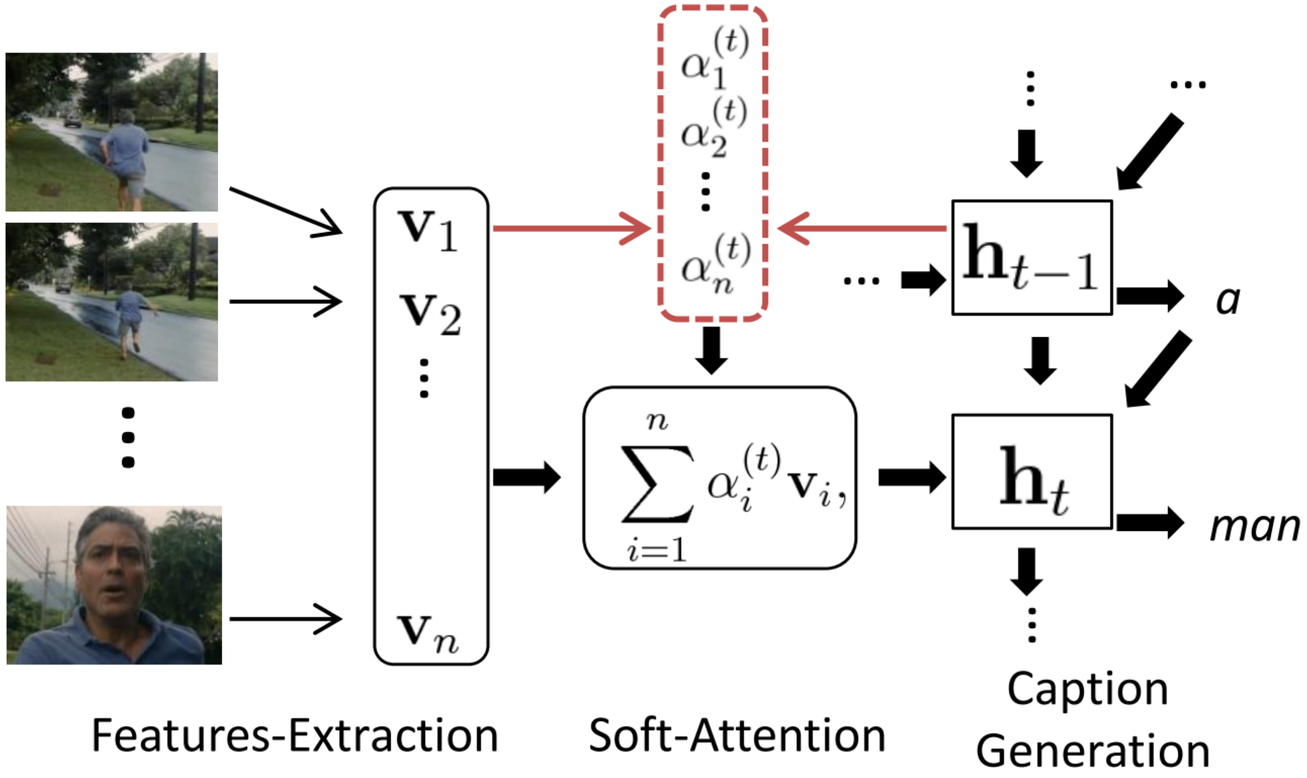
Résultats

Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

Pour vidéos

On peut aussi le faire sur des **vidéos**!



Li et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*

Génération de légende

Résultats

+Local+Global: A **man** and a **woman** are **talking** on the **road**

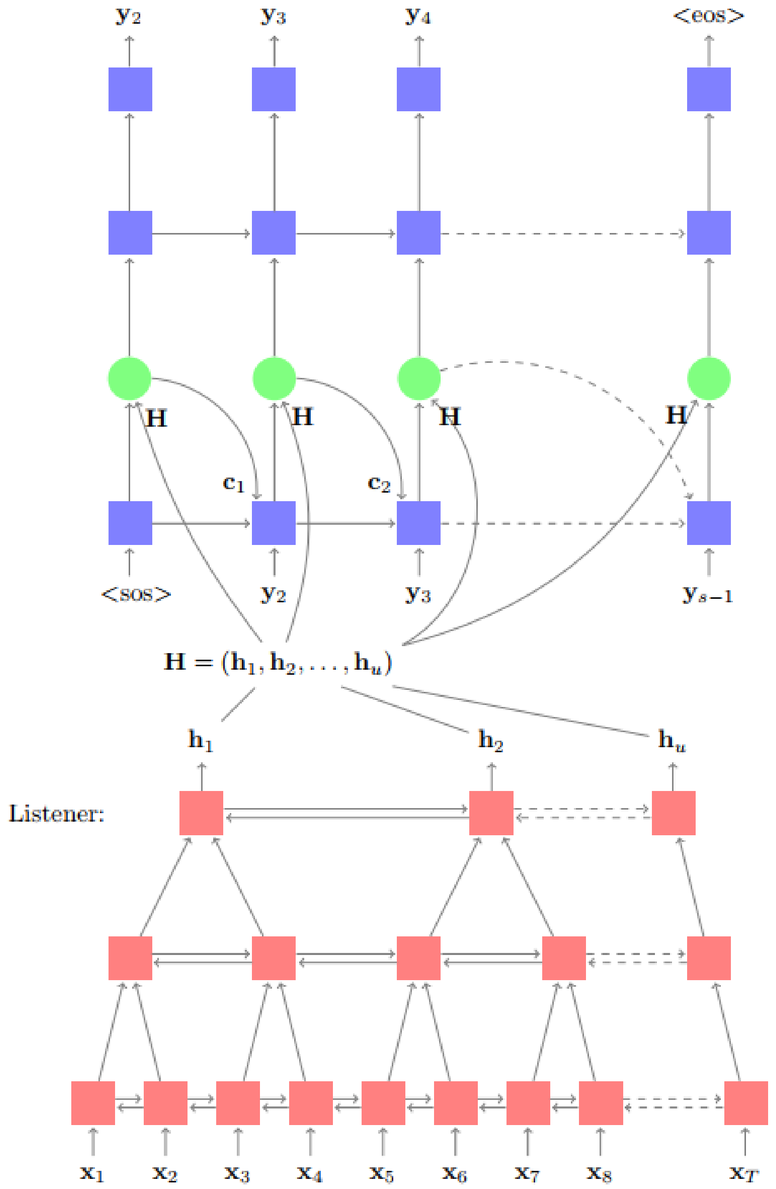
Ref: A man and a woman ride a motorcycle

Reconnaissance de la parole

Modèle « Listen Attend and Spell »

Système complet de reconnaissance de la parole.

- Entrée: Spectrogramme
- Sortie: 29 Caractères (26 lettres + espace + <SOS> + <EOS>)
- 3 Parties: Listener, Attender and Speller



Chan et al., *Listen, Attend and Spell*

Image from https://github.com/v0lta/masterThesis/blob/master/thesis.pdf

Reconnaissance de la parole

Modèle « Listen Attend and Spell »

Système complet de reconnaissance de la parole.

- Listener: Traîte la séquence d'entrée
- Attender: Choisit où écouter dans la séquence traîtée
- Speller: Génère les caractères

B-LSTMs, LSTMs, Attention

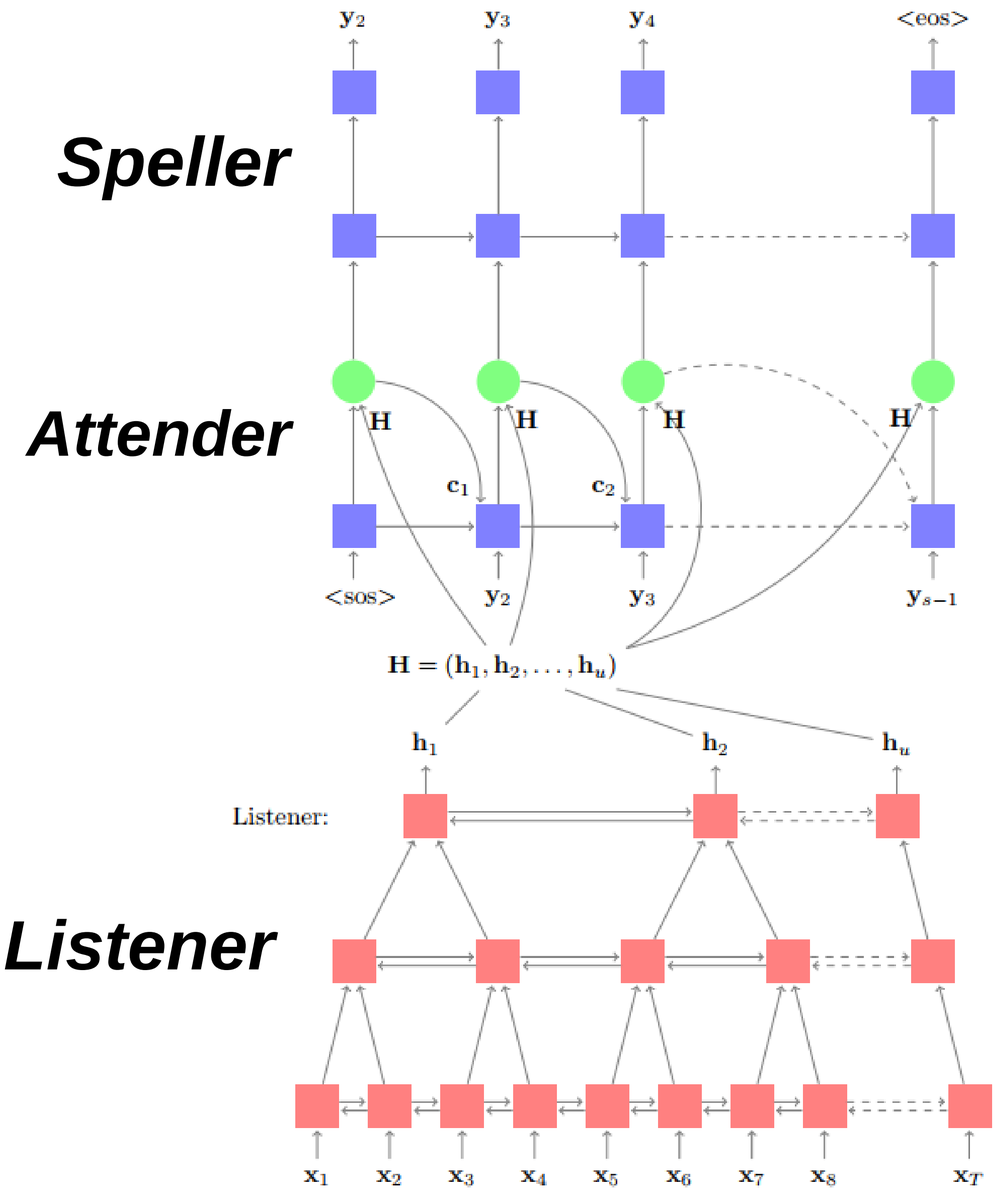
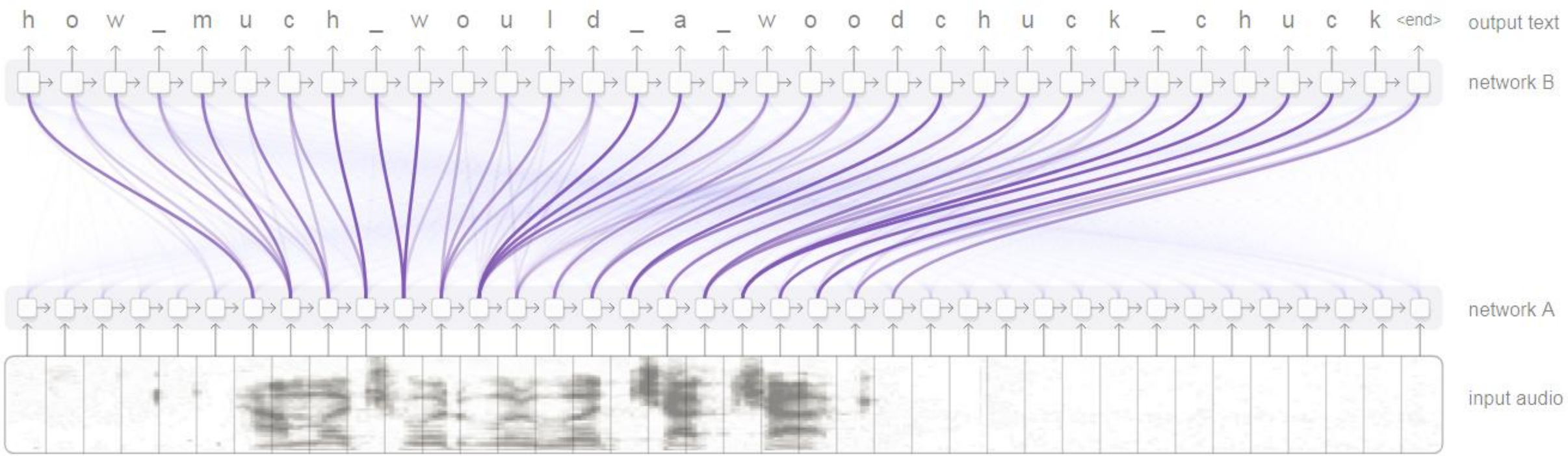


Image: https://github.com/v0lta/masterThesis/blob/master/thesis.pdf

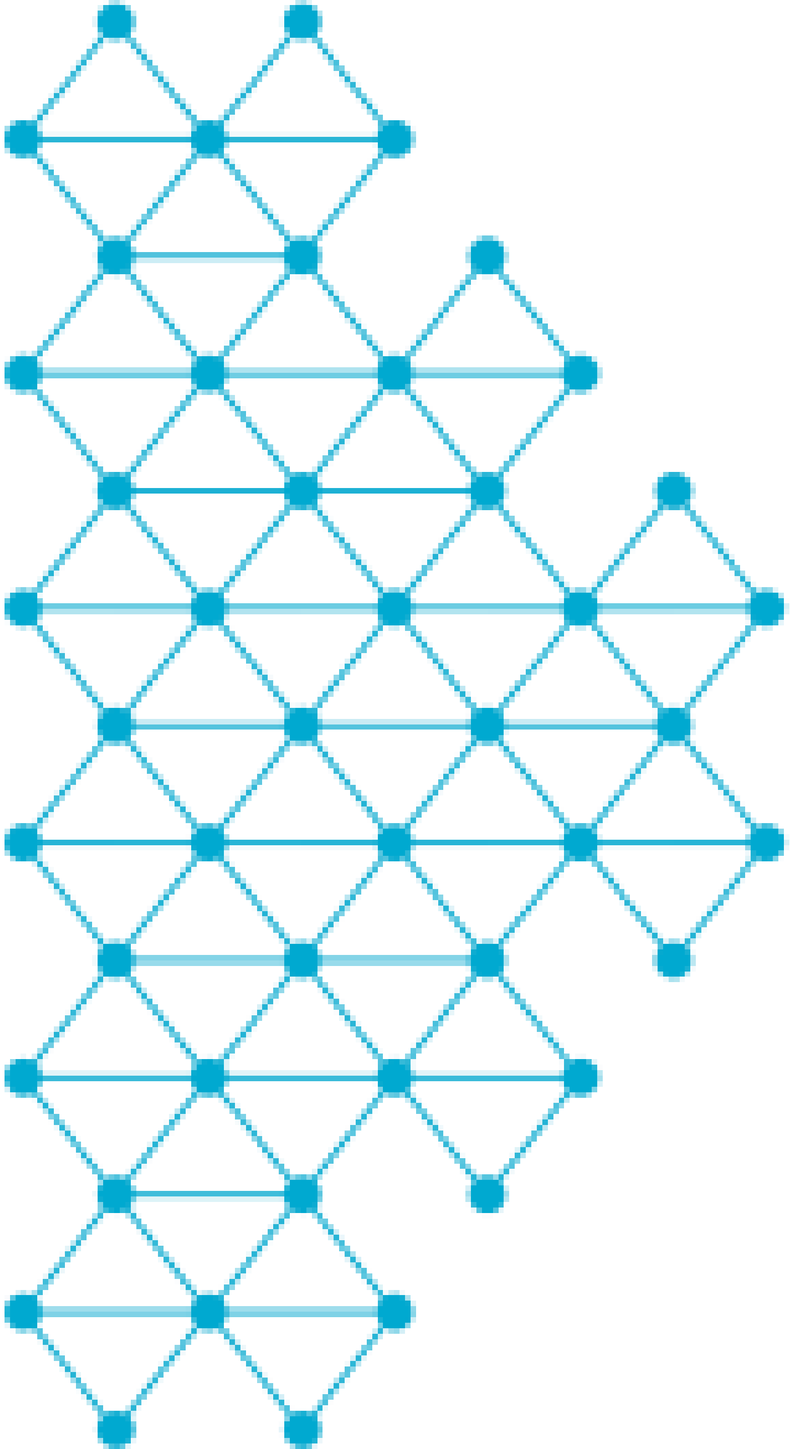
Reconnaissance de la parole

Visualisation de l'attention



Les traits mauves montrent où le RNN B (Attender) porte son attention dans la séquence générée par le RNN A (Listener).

Image from [Christopher Olah's blog]



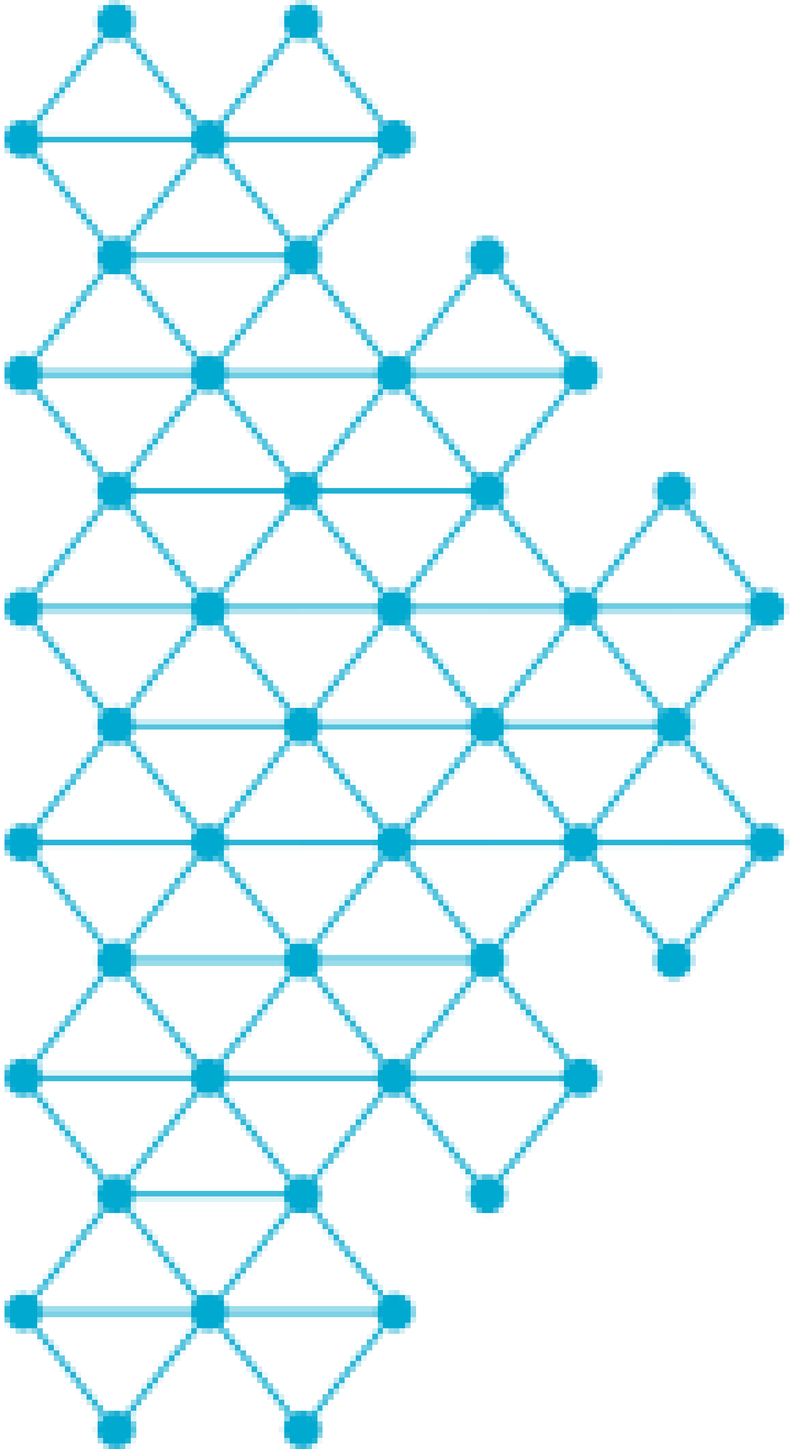
4.Librairies et Références

RNNs dans les librairies

- Les RNNs sont présents dans les librairies de Deep Learning!
  - Tensorflow: https://www.tensorflow.org/tutorials/recurrent
  - Torch7: https://github.com/jcjohnson/torch-rnn
  - Blocks (Theano): http://blocks.readthedocs.io/en/latest/rnn.html
  - Keras : https://keras.io/layers/recurrent/
  - Etc…

Références

- Blog de Christopher Olah sur les LSTMs: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Blog de Christopher Olah sur les mécanismes d'attention: https://distill.pub/2016/augmented-rnns/
- Blog de Andrej Karpathy sur les RNNs: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- The Deep Learning Book (Goodfellow et al.): http://www.deeplearningbook.org/



Questions ?