Apprentissage statistique : modélisation décisionnelle et apprentissage profond (RCP209)

Réseaux de neurones récurrents

Nicolas Audebert
nicolas.audebert@lecnam.net
http://cedric.cnam.fr/vertigo/Cours/ml2/

Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

17 mai 2023

Plan du cours

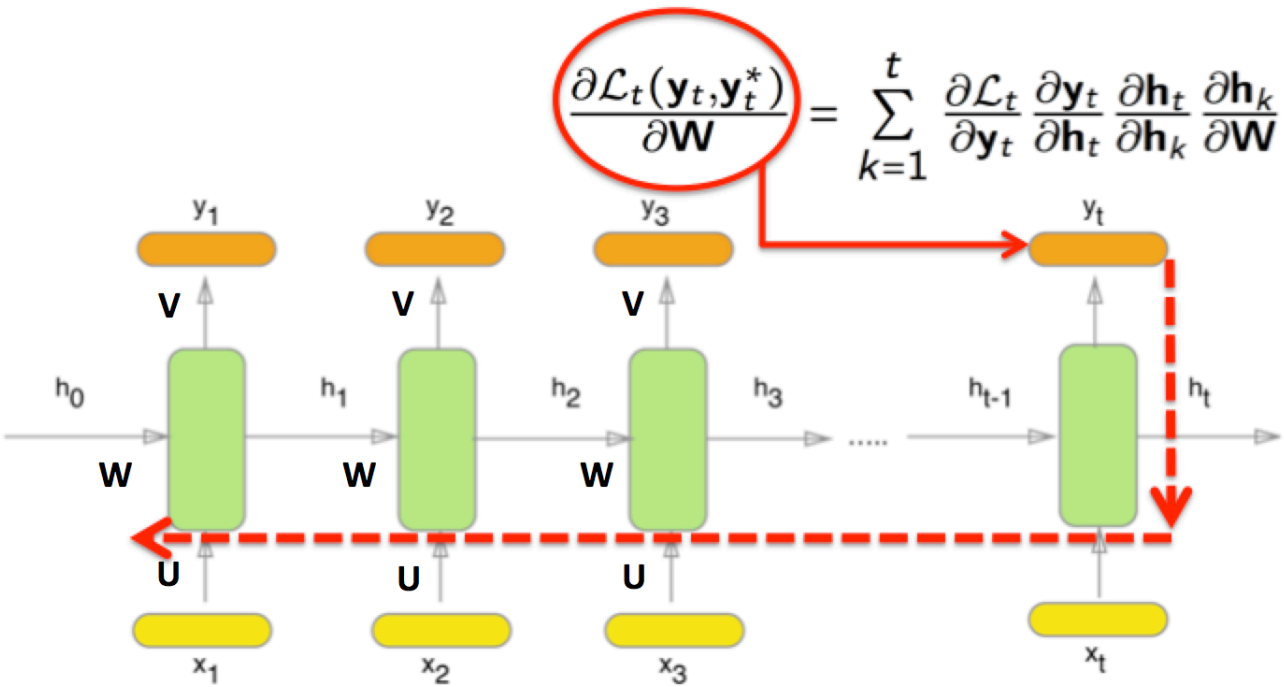
Rappel : *Back-Propagation Through Time* (BPTT)

État interne de la cellule récurrente : $\boxed{\mathbf{h}_t = \tanh(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b}_h)}$

Sortie au pas de temps t : $\boxed{\mathbf{y}_t = \mathrm{softmax}(\mathbf{V}\mathbf{h}_t + \mathbf{b}_y)}$

- Les paramètres $\mathbf{W}, \mathbf{U}, \mathbf{V}$ sont **partagés** pour tous les pas de temps $\Longrightarrow$ les gradients dépendent de tout l'historique de la séquence.
- Exemple : pour $\mathbf{W}$, on cherche le gradient $\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}}$:

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}}$$



Analyse du gradient

- Gradient pour les paramètres $\mathbf{W}$: $\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \boxed{\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}} \frac{\partial \mathbf{h}_k}{\partial \mathbf{W}}$
- Par *chain rule* : $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}$
- L'état interne s'obtient par application d'une activation non-linéaire f, par exemple : $\mathbf{h}_t = \tanh(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b}_h)$
- On obtient la matrice jacobienne : $\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} = \mathbf{W}^T \operatorname{diag}[f'(\mathbf{h}_{j-1})]$

$\Rightarrow$ comment se comporte $\boxed{\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \| = \| \prod_{j=k+1}^{t} \mathbf{W}^T \operatorname{diag}[f'(\mathbf{h}_{j-1})] \|}$?

BPTT : gradients explosifs et évanescents

Comportement du gradient d'un RNN

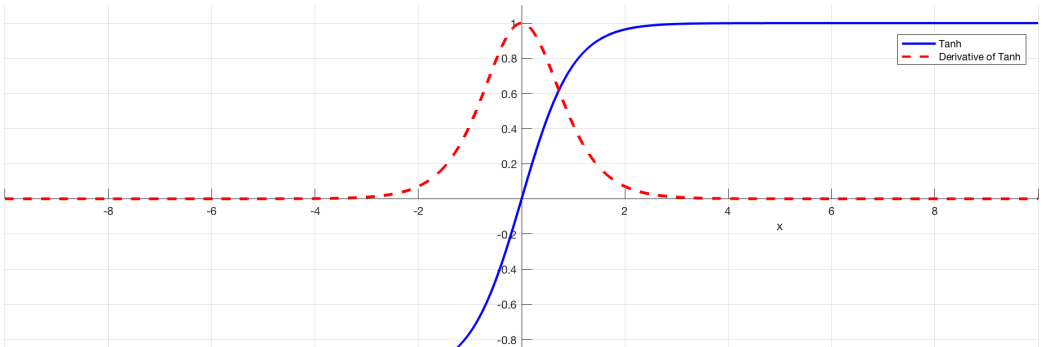
$$\|\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}\| = \|\prod_{j=k+1}^{t} \mathbf{W}^T \operatorname{diag}[f'(\mathbf{h}_{j-1})]\| \leq (\beta_w \beta_h)^{t-k}$$

- β_h dépend de l'activation ($\tanh \rightarrow \beta_h = 1$, $\sigma \rightarrow \beta_h = 0,25$)
- β_w dépend de la plus grande valeur propre de $\mathbf{W}$

On distingue deux cas :

- Si $\beta_h \cdot \beta_w > 1$, alors les gradients $\nearrow$ à chaque t $\Longrightarrow$ *exploding gradients*
- Si $\beta_h \cdot \beta_w < 1$, alors les gradients $\nearrow$ à chaque t $\Longrightarrow$ *vanishing gradients*

Cette analyse est vraie pour tous les réseaux profonds mais est exacerbée par la "profondeur"des RNN !



Solutions anti-gradients explosifs

- Troncature de la BPTT ($\ominus$ capacité limitée à modéliser des dépendances temporelles)
- Méthodes de régularisation sur les poids, par exemple $\|W\|_2$
- *Gradient clipping* : tronquer les gradients
  - Pour un gradient $\mathbf{g}$ et un seuil τ, si $\|\mathbf{g}\| > \tau$, alors $\mathbf{g} \leftarrow \frac{\tau}{\|\mathbf{g}\|}\mathbf{g}$.

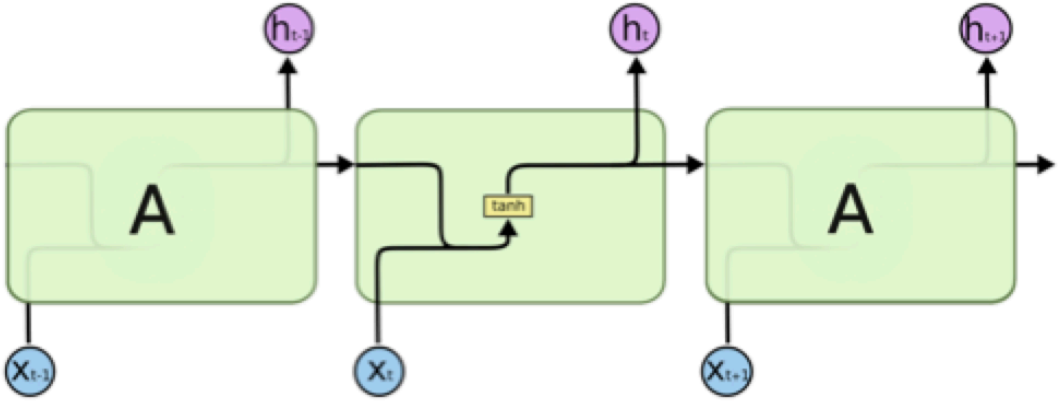
Les gradients explosifs sont relativement faciles à détecter et à corriger.

Solutions anti-gradients évanescents

- Troncature de la BPTT ($\ominus$ capacité limitée à modéliser des dépendances temporelles)
- Méthodes spécifiques de régularisation pour contrôler l'amplitude des gradients
- Activation ReLU plutôt que $\tanh$ ou sigmoïde
- Optimiseurs de quasi-second ordres (*Hessian-free* + *damping*) ($\ominus$ plus lent, plus coûteux)
- Architectures de cellules récurrentes spécifiques $\Longrightarrow$ voir la suite !

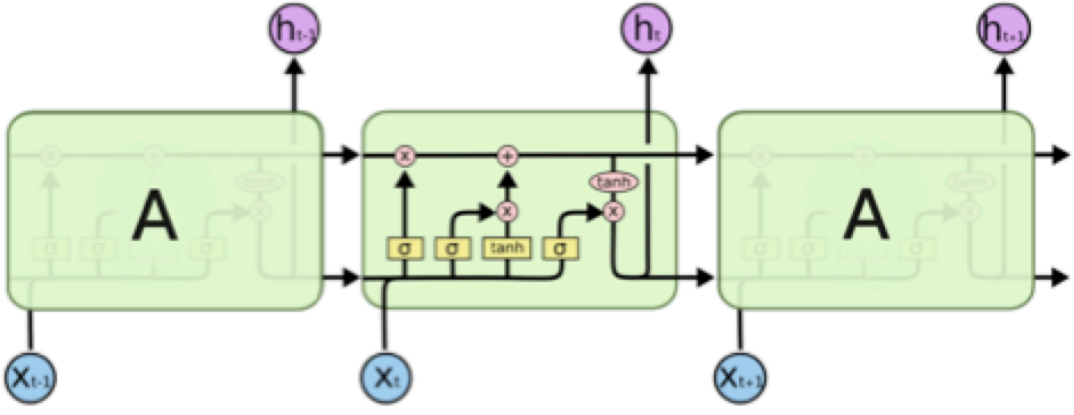
LSTM : *Long Short-Term Memory*

Cellule récurrente classique



$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

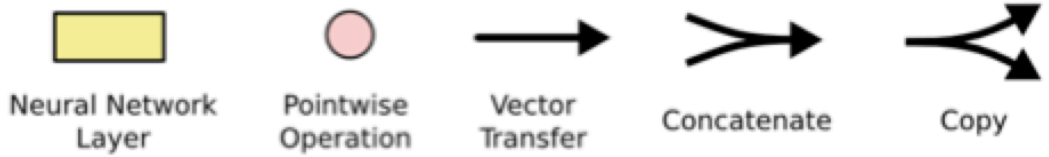
Long Short-Term Memory (LSTM) Hochreiter et Schmidhuber 1997



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

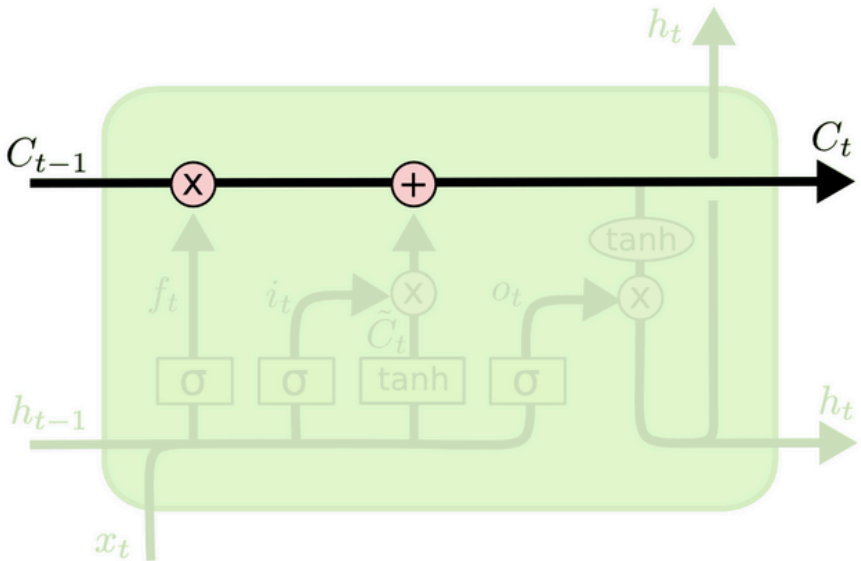
$$h_t = o \odot \tanh(c_t)$$



Credit: C. Olah

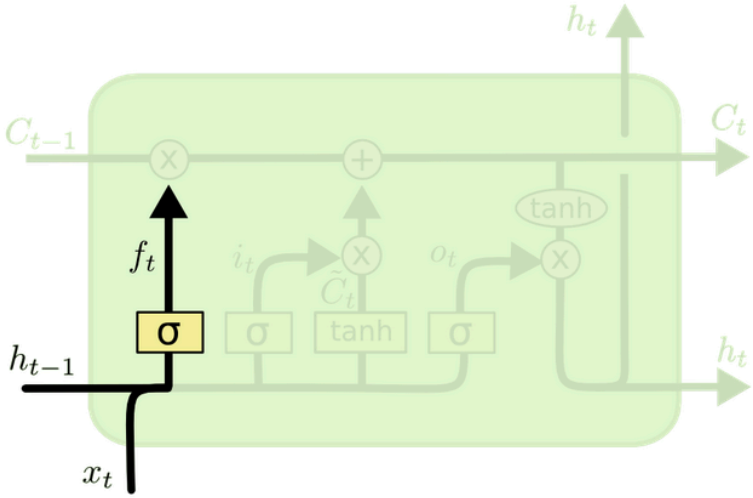
Architecture LSTM

- Modification principale : ajout d'un état de la cellule $\mathbf{C_t}$ passé directement au t suivant
- Simple à implémenter, permet à l'information (gradient) d'avoir un flux non-interrompu selon le chemin $\mathbf{C_t}$
- La cellule LSTM ajoute ou retire de l'information de son état interne $\mathbf{C_t}$



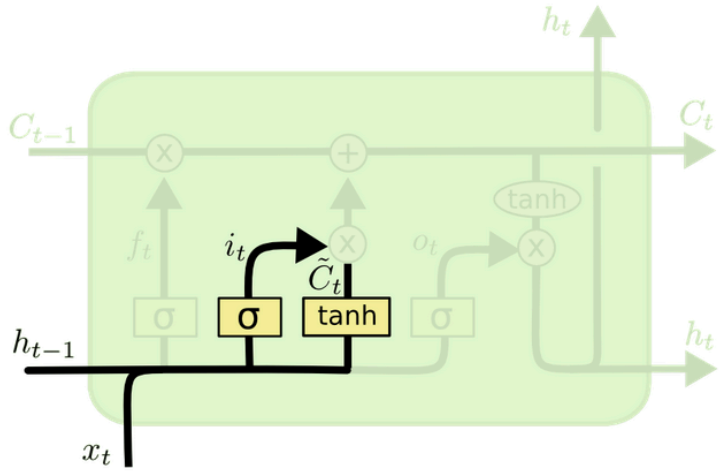
LSTM : portes d'oubli et d'entrée

- **Porte d'oubli** (*forget gate*) $\mathbf{f_t}$: contrôle la conservation ou l'effacement de la cellule



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- **Portée d'entrée** (*input gate*) $\mathbf{i_t}$: contrôle si la cellule est écrite ou non
- $\tilde{\mathbf{C}}_\mathbf{t}$: ce qui est écrit dans la cellule
  - $\sigma \in [0, 1]$ (contrôle/commutateur), $\tanh \in [-1, 1]$ (non-linéarité)

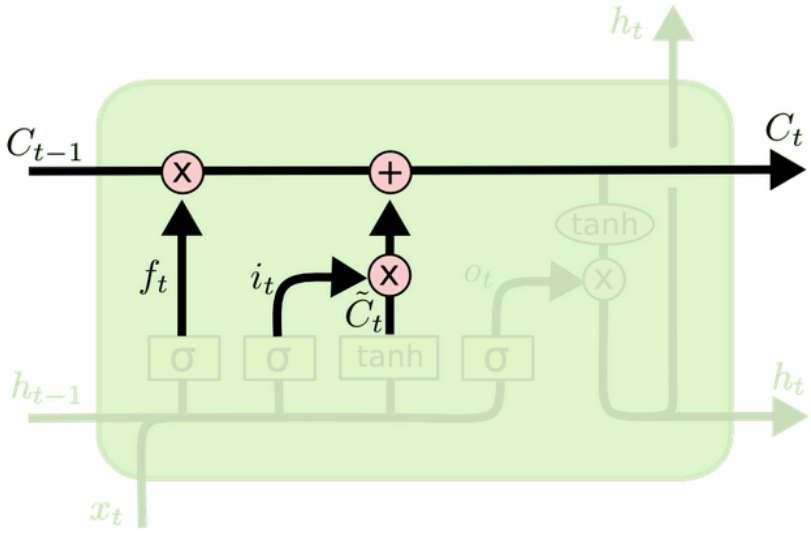


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

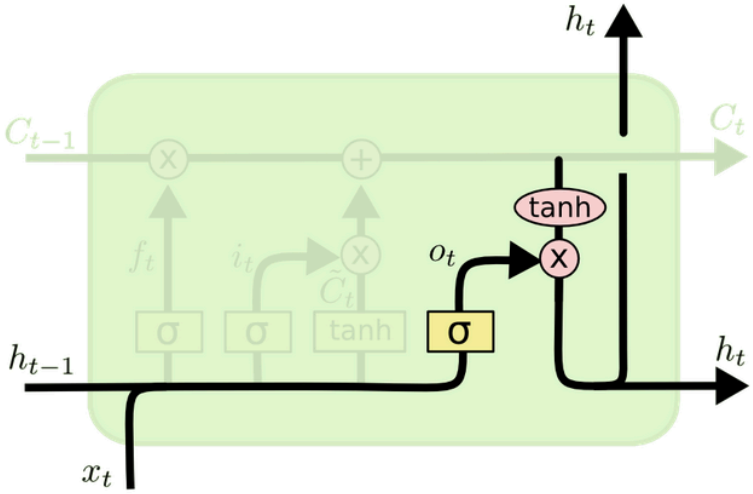
LSTM : mise à jour et de porte de sortie

- **Mise à jour** : efface la cellule si $f_t \approx 0$, ajoute le contenu de $\mathbf{i_t} \times \mathbf{\tilde{C}_t}$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- **Porte de sortie** $\mathbf{o_t}$ contrôle la proportion de l'état interne précédent remplacé par l'état actuel de la cellule

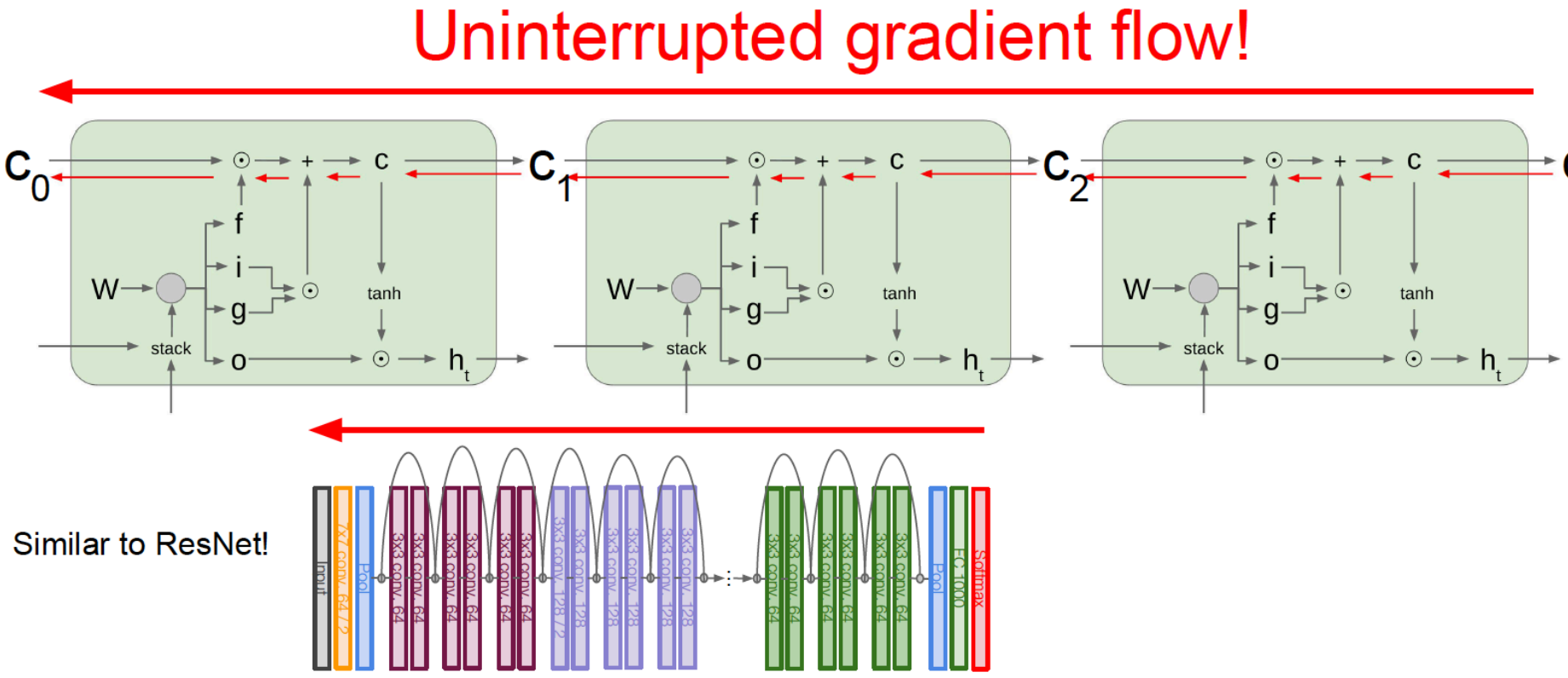


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

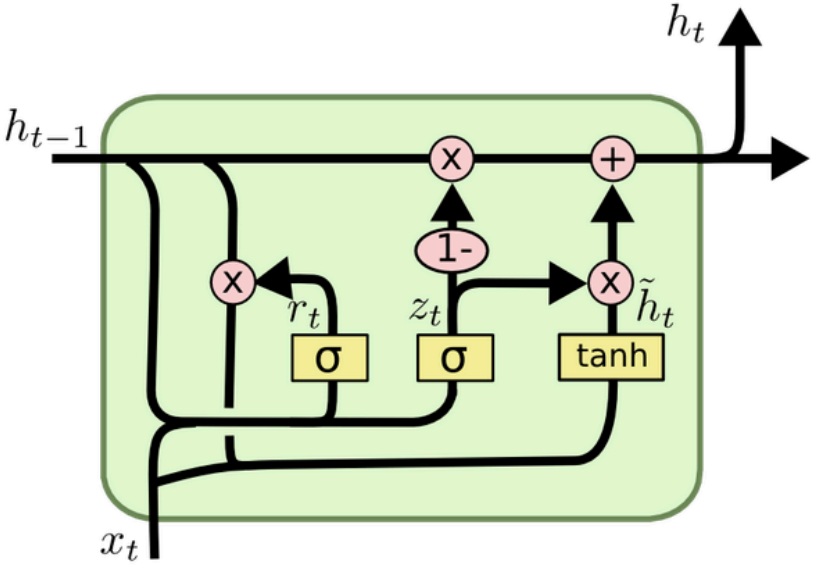
Flot du gradient

- Multiplications et additions élément par élément, pas de multiplication matricielle par $\mathbf{W}$:



GRU (*Gated Recurrent Unit*)

- GRU Cho et al. 2014 : Variante simplifiée des LSTM avec moins de paramètres
  - Combine les portes d'oubli et d'entrée dans une porte de *mise à jour*
  - Fusionne l'état de la cellule et l'état caché interne



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

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

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

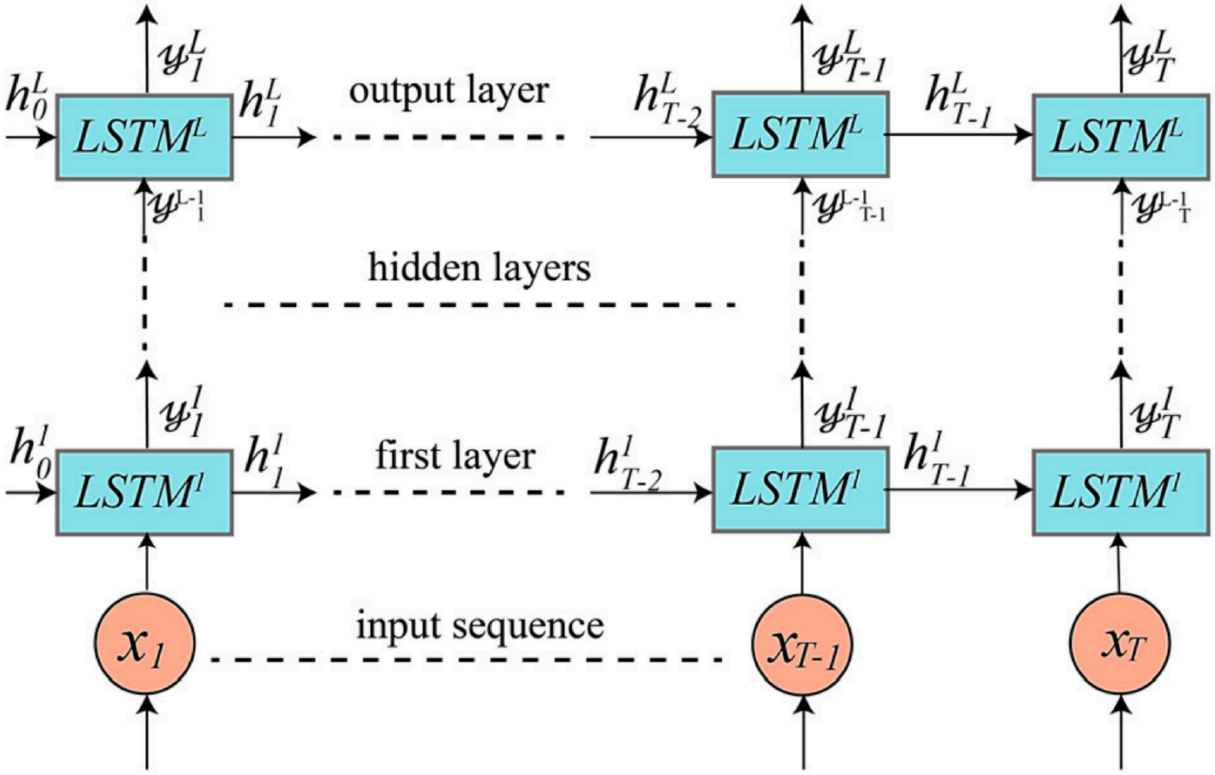
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

⊕ Plus simple que les LSTM, plus rapide, avec moins de paramètres

⊖ Performances généralement inférieures à architecture équivalente

RNN profonds

- Empiler des couches RNN/LSTM/GRU : apprendre des représentations de complexité ↗
- "Deep LSTM" : architecture performante pour modéliser des problèmes séquentiels avec beaucoup de données

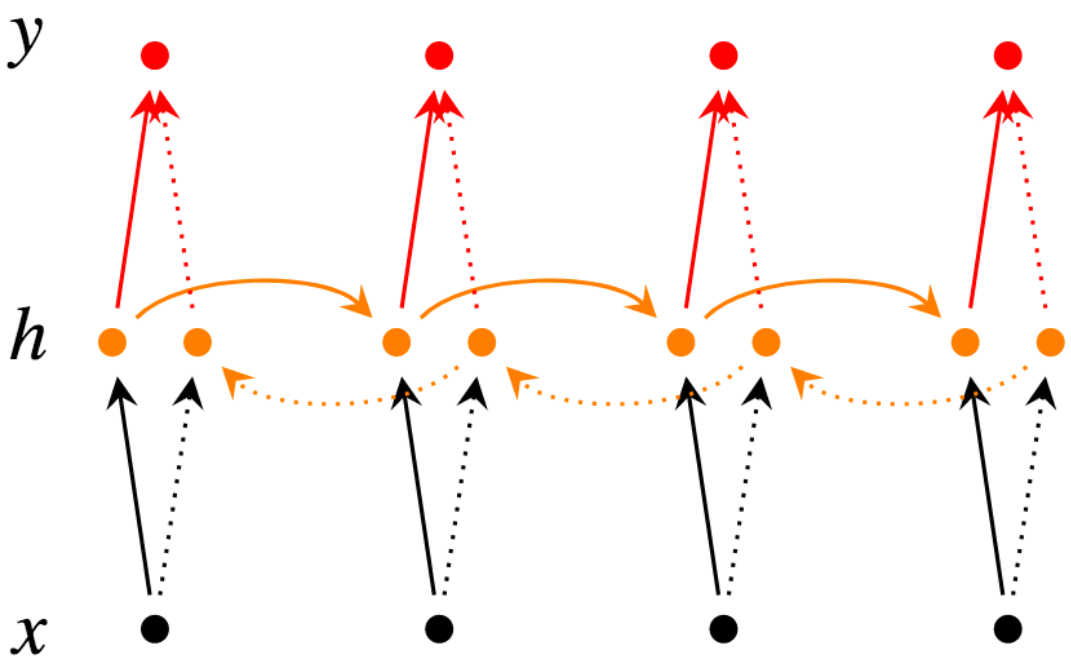


RNN bi-directionnels

Pour certains problèmes, il n'est pas nécessaire d'avoir l'hypothèse de causalité : on peut avoir besoin d'examiner x_{t+1} pour prendre une décision concernant x_t.

- Par exemple, pour la traduction automatique de textes, on souhaite incorporer de l'information concernant les mots qui précèdent et qui succèdent au mot à traduire.

Solution : appliquer en parallèle deux RNN, l'un dans le sens $t \nearrow$, l'autre dans le sens $t \searrow$ et concaténer les états internes :



$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\vec{h}_t; \overleftarrow{h}_t] + c)$$

$h = [\vec{h}; \overleftarrow{h}]$ now represents (summarizes) the past and future around a single token.

Plan du cours

Représentations *one-hot*

Comment représenter un texte (séquence de mots) au format numérique ?

Définitions

- Vocabulaire $|V|$: ensemble des *tokens* ($\sim$ mots ou caractères) uniques apparaissant dans le corpus ($\sim$ dictionnaire)
- Corpus : ensemble d'apprentissage $\mathcal{X} = (\{x_1\}_t, \{x_2\}_t, \ldots, \{x_n\}_t)$ ($\{x_i\}_t$ phrase = séquences de mots)

Encodage *one-hot*

- Encodage le plus simple des données textuelles
- Vecteur binaire de longueur égale à la taille du vocabulaire $|V|$
- $x[k] = 1$ si x représente le *token* k, 0 sinon
- $|V|$ petit pour les caractères alphanumériques (≈ 30), grand pour des mots ($\sim 10^4$)
- $v \in V$ est un *token* (élément, mot ou caractère)

Exemple : $V = \{''a'', '' b'', '' c'', \ldots\}$, $''a'' = (1, 0, 0, \ldots, 0)$, $''b'' = (0, 1, 0, \ldots, 0)$

Modèle sac-de-mots

- Séquence de *tokens* : vecteur $|V|$ du nombre d'occurence de chaque *token*

Représentation sac-de-mots

- Descripteur textuel encore utilisé avec des modèles décisionnels "shallow"
- ⊕ Simple à implémenter, taille fixe quelque soit la longueur de la séquence
- ⊕ Compétitif pour certaines tâches en traitement automatique du langage (par exemple, classification du sujet d'un texte, identification de la langue)
- ⊖ Ne prend pas en compte l'ordre de la séquence (mais peut être étendu avec un vocabulaire de n-grammes)
- ⊖ Taille très grande pour des vocabulaires non-triviaux

Représentation *one-hot* et similarité

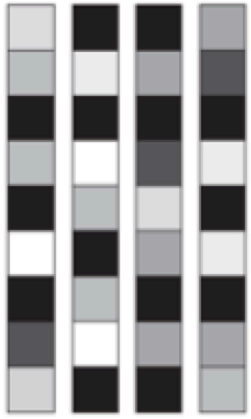
- Limite des vecteurs one-hot : $\langle\ r(\text{"}motel\text{"})\ ;\ r(\text{"}hotel\text{"})\ \rangle = 0$

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

- Apprentissage de représentation : apprendre un plongement lexical (*text embedding*) dont la similarité cosinus reflète la similarité sémantique entre les *tokens*

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded

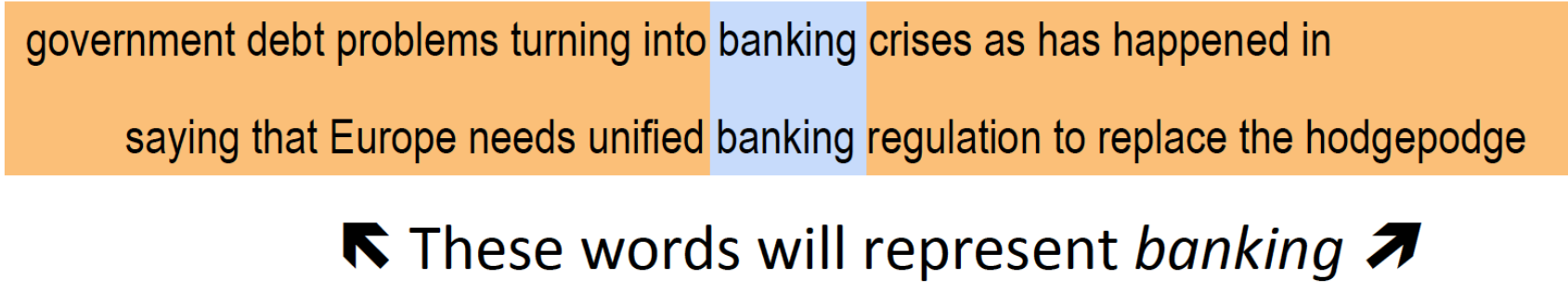
VS



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Word embeddings

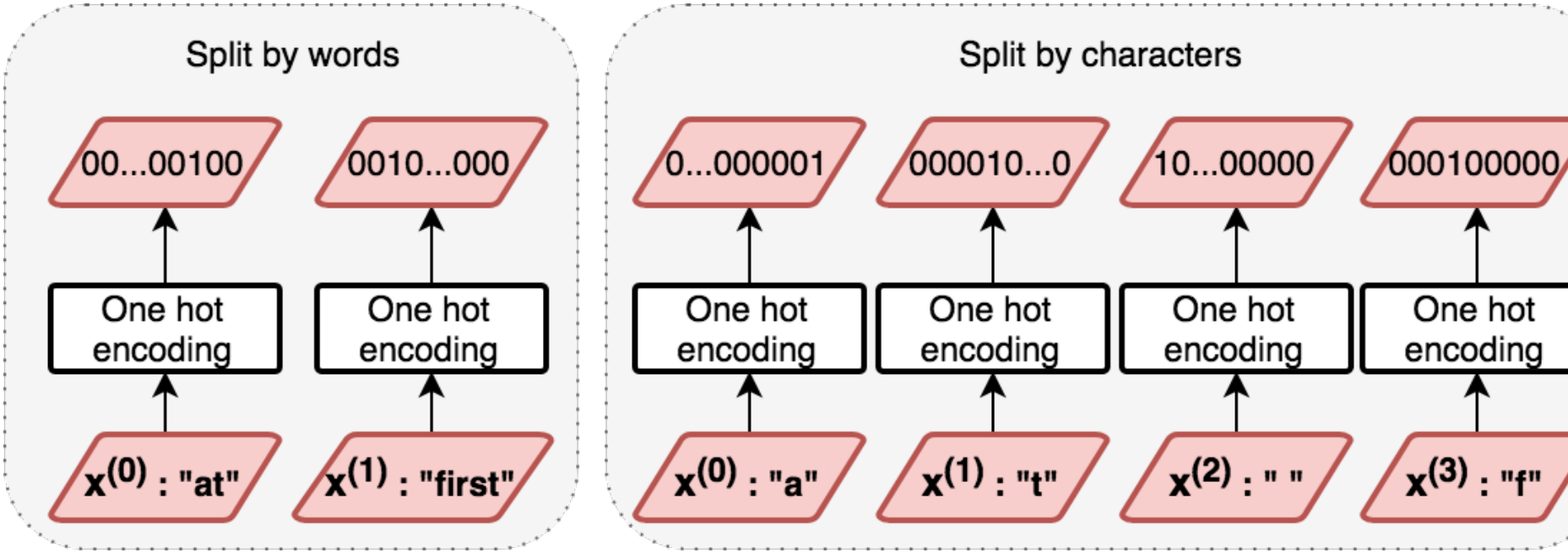
- Apprendre une projection de la représentation *one-hot* vers un espace vectoriel de dimension plus faible
- Idée générale : un mot se représente à partir d'une combinaison de ses voisins (hypothèse distributionnelle)



- Différentes approches :
  - Word2vec Mikolov et al. 2013 : MLP à une couche cachée, apprend à identifier un mot à partir de ses 5 voisins avant et après
    - Couche d'entrée : one-hot $\rightarrow$ embedding
    - Couche de sortie : embedding $\rightarrow$ classification
  - $\approx$ GloVe Pennington, Socher et Manning 2014 : décomposition en valeurs singulières de la matrice des co-occurrences
  - BERT Devlin et al. 2019 : architectures Transfromers $\Rightarrow$ RCP217

RNN pour le traitement automatique de la langue

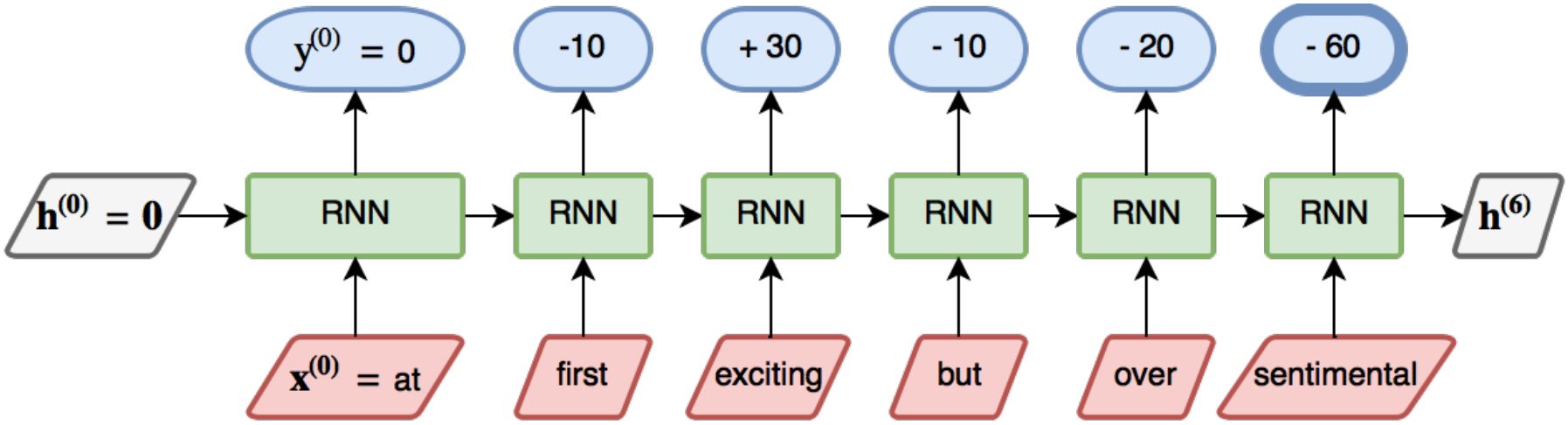
1. Extraire les entrée textuelles sous forme de *tokens* (caractères ou mots)
2. Encodage *one-hot* des *tokens*



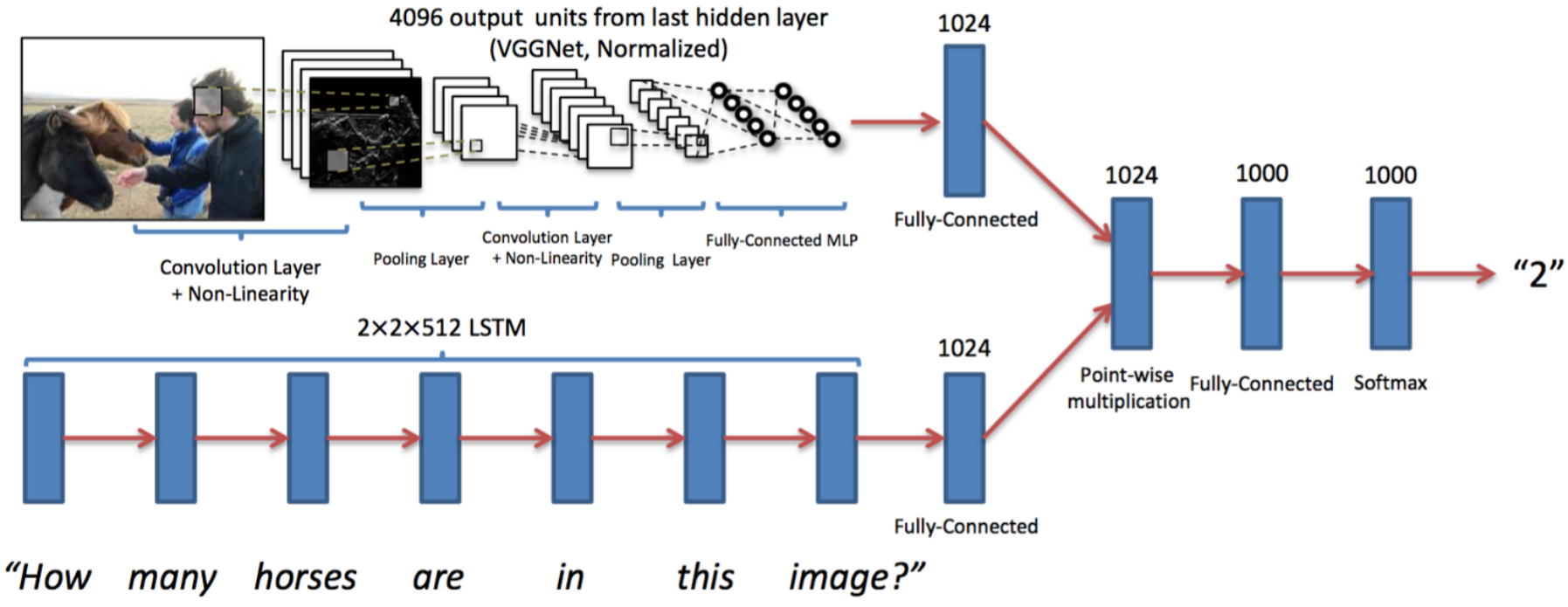
3. Division du texte sous forme de séquence “temporelle”
4. Entraînement d’un RNN sur la séquence
   - Optionnel : ajouter un MLP pour apprendre un *embedding* après l’encodage *one-hot*

Many-to-one : classification de texte

- Exemple : classification de sentiment
  - Entrée : "`At first exciting but over sentimental`"
    - *Token* $\Leftrightarrow$ mot
  - Sortie : score y (par exemple -60 = critique négative)

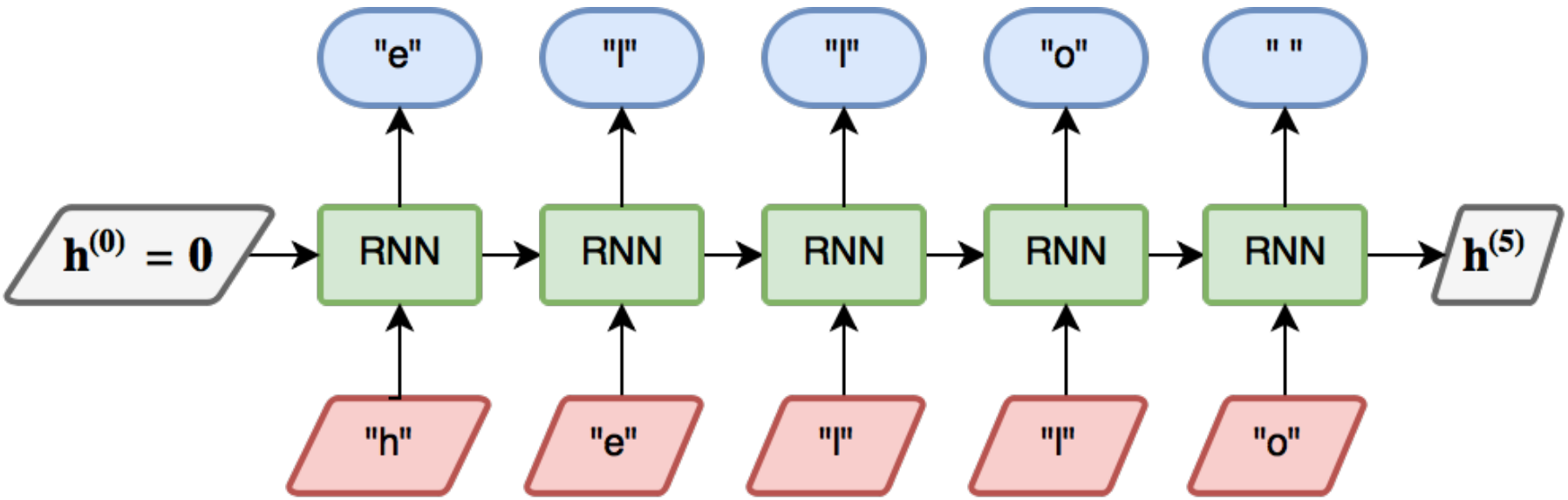


Many-to-one : VQA



Many-to-many : génération de séquence

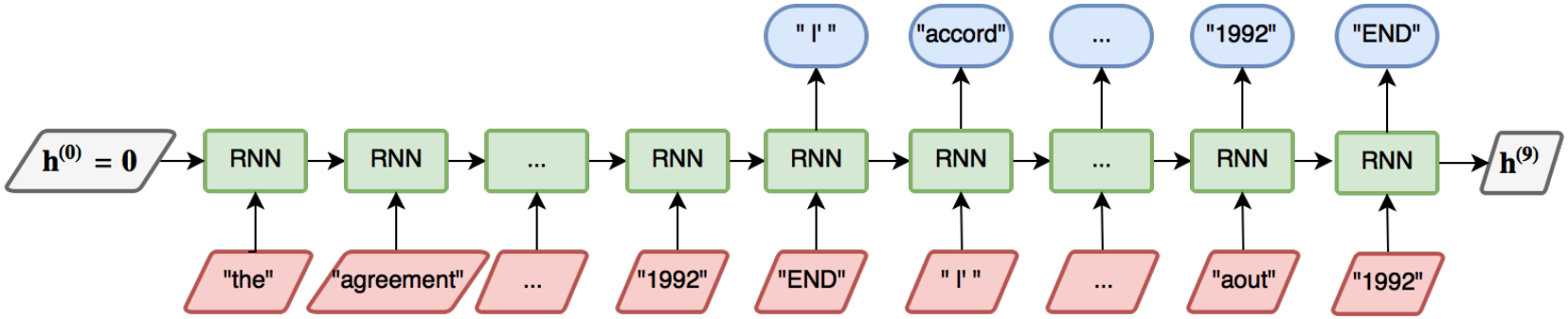
- Génération de texte, de musique, etc.
  - Entrée : séquence de caractères
  - Sortie : prédiction du caractère suivant $y = x_{t+1}$
- En théorie, problème *many-to-many* parallèle
- En pratique, appris avec une BPTT tronquée $\Longrightarrow$ *many-to-one* : prédire le caractère suivant à partir des k caractères précédents



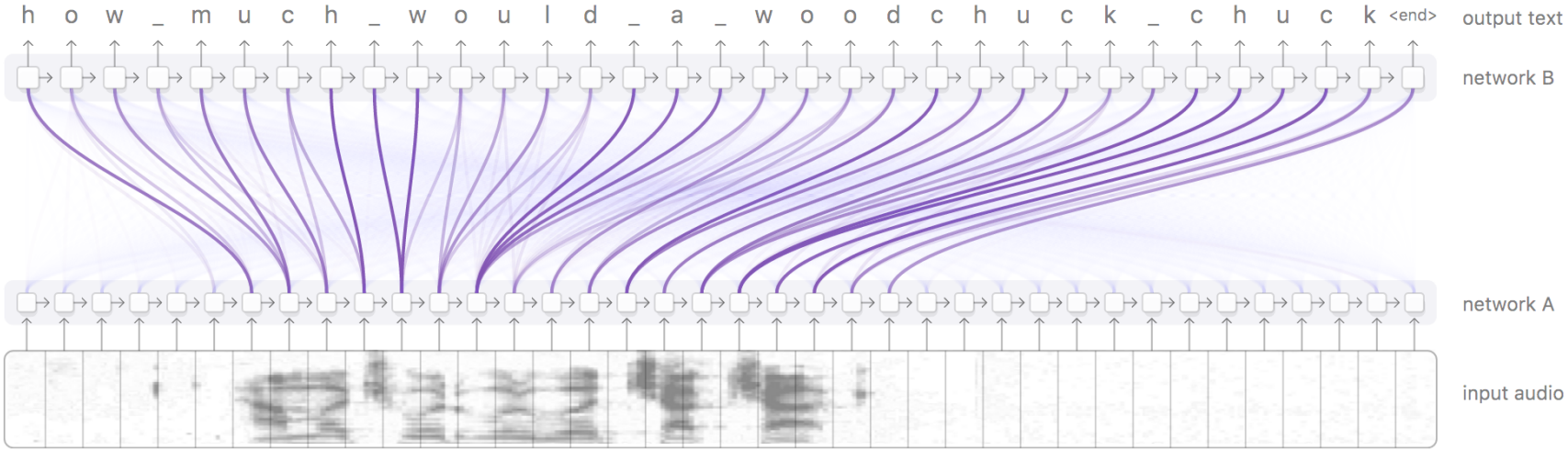
Many-to-many : séquence vers séquence

Modèles Seq2Seq

- Traduction automatique
  - Entrée : "`The agreement on the European Economic Area was signed in August 1992.`"
  - Sortie : "`L'accord sur la zone économique européenne a été signé en août 1992.`"



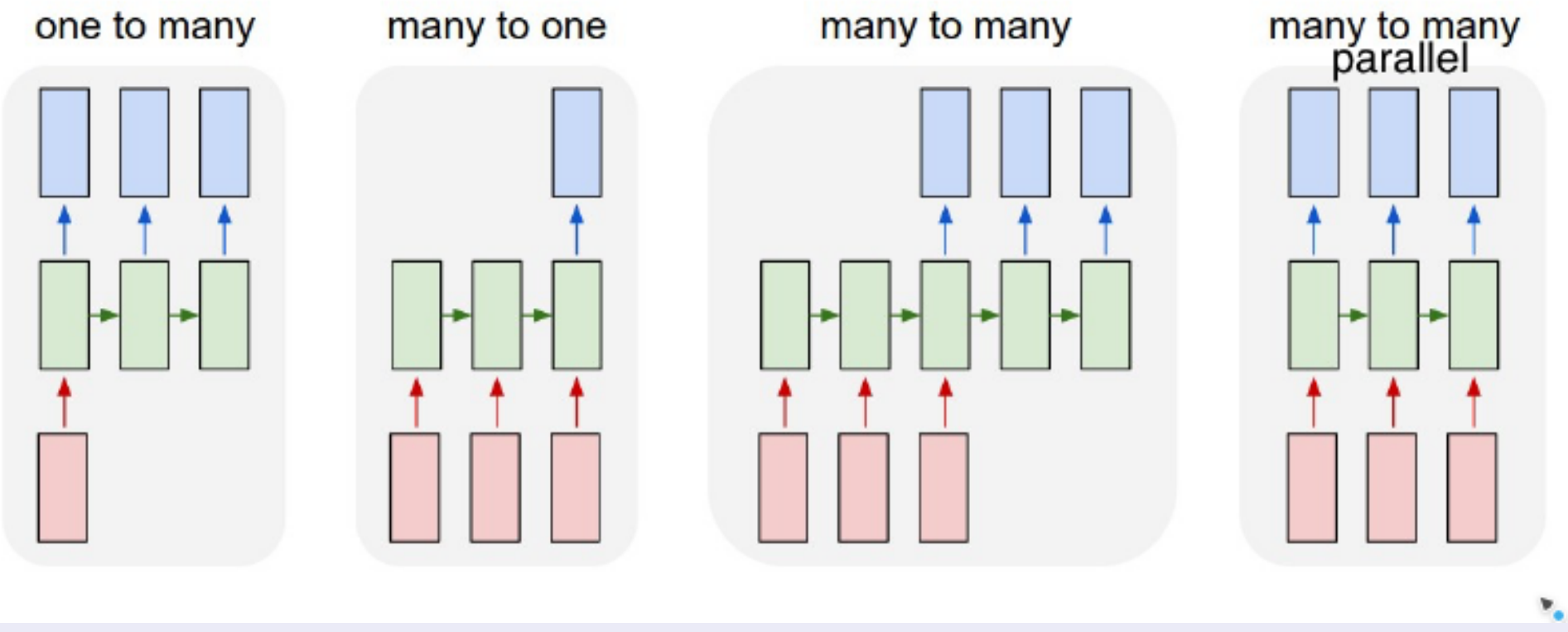
- Transcription (*speech2text*)
  - Entrée : forme d'onde audio/spectrogramme
  - Sortie : "`How much would a woodchuck chuck`"



Plan du cours

Modélisation décisionnelle sur des séquences

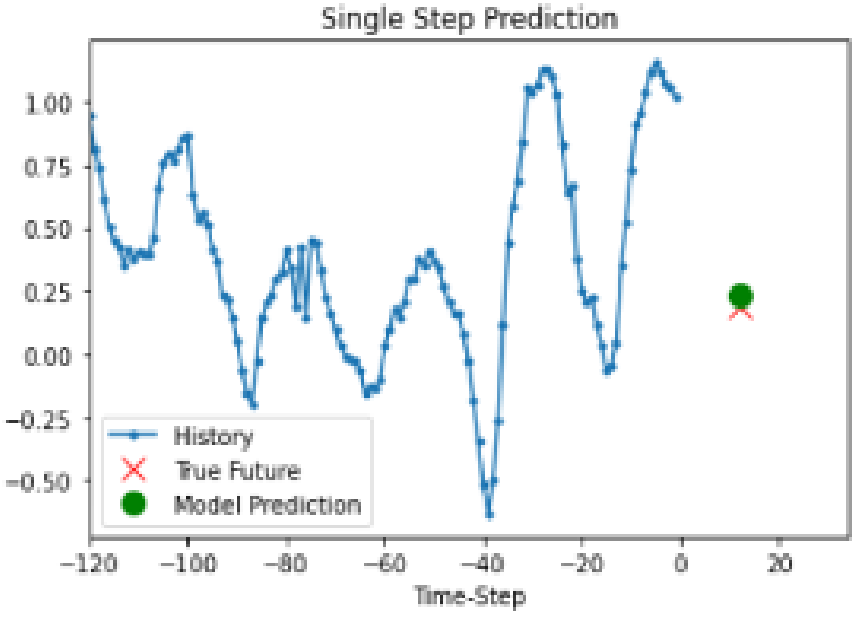
Différents types de problèmes

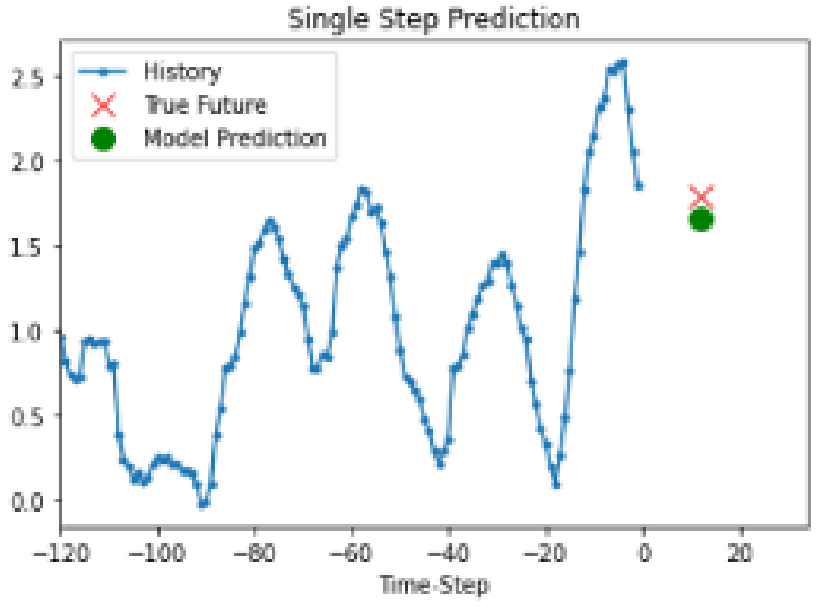


Prédictions de séquences

Prédiction de séries temporelles ⇒ voir la séance de travaux pratiques

- Prédiction de séries temporelles mono-variées ou multi-variées
- Comparaison cellule RNN classique et GRU/LSTM





En général, problème *many-to-many*.

- Classification ou régression d'une seule valeur à partir d'une série temporelle ⇒ *many-to-one*.

Données spatio-temporelles

Manipulation de données spatiales et temporelles (par exemple, vidéo) : tenseurs 4D.
Architecture hybrides mêlant CNN et RNN, par exemple ConvLSTM Shi et al. 2015 :

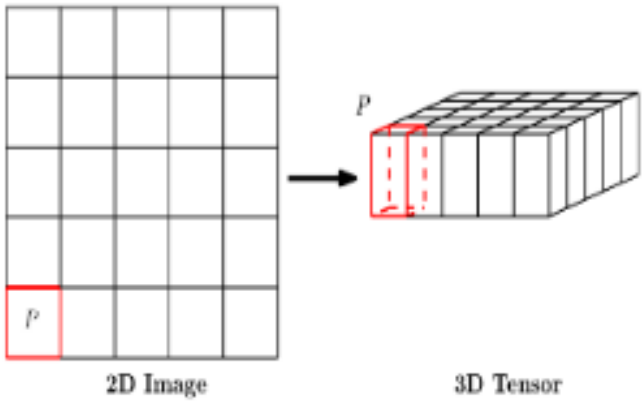


Figure 1: Transforming 2D image into 3D tensor

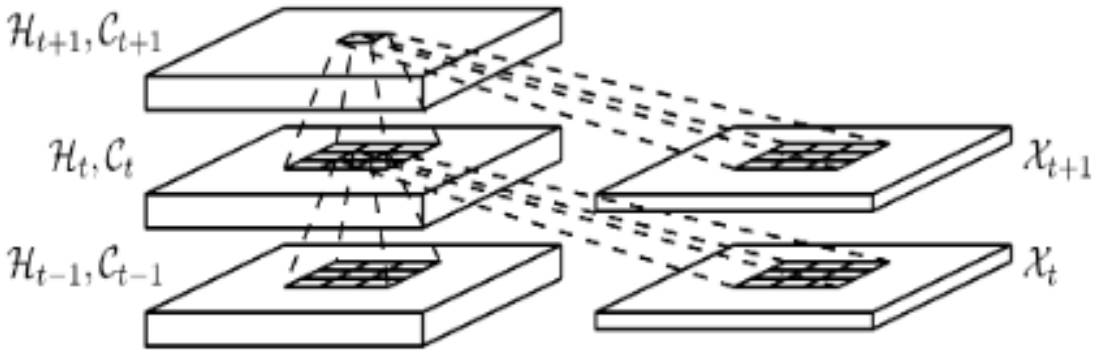


Figure 2: Inner structure of ConvLSTM

One-to-many : description d'images

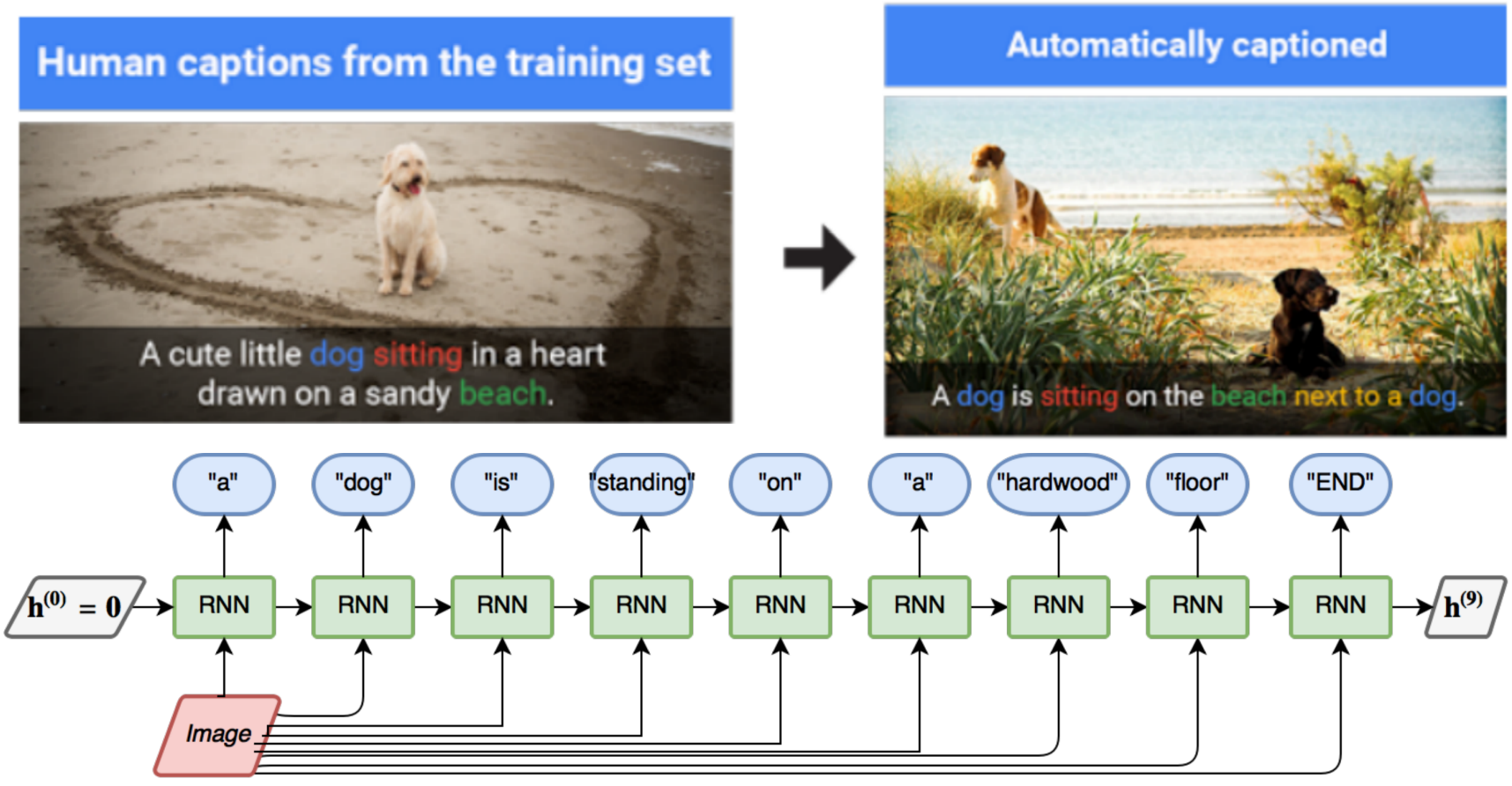
- Entrée : image ($W \times H \times C$)
- Sortie : légende (phrase de T mots en langage naturel)

Approches inspirées des travaux en traduction automatique : encodeur image (CNN) + décodeur texte (RNN)

One-to-many : description d'images

- Entrée : image ($W \times H \times C$)
- Sortie : légende (phrase de T mots en langage naturel)

Approches inspirées des travaux en traduction automatique : encodeur image (CNN) + décodeur texte (RNN)



Many to one : Visual Question Answering (VQA)

Goal : build a system that can answer questions about images

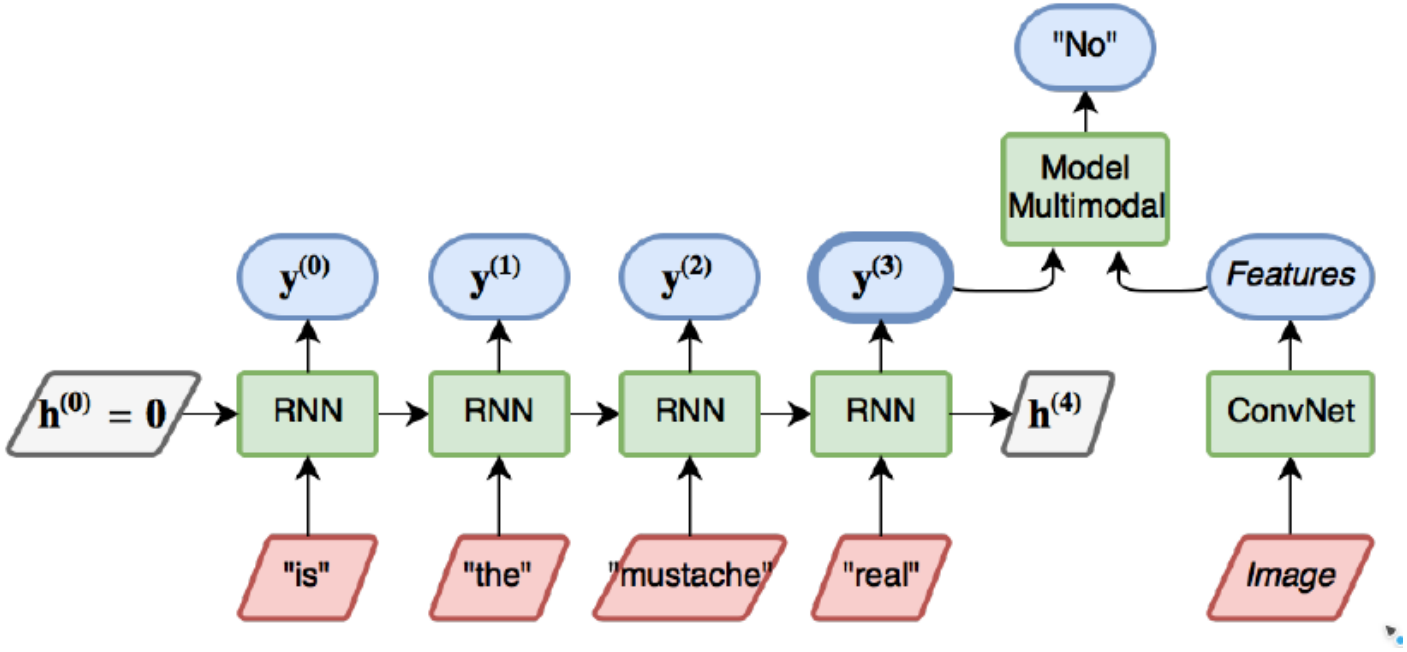
How many slices of pizza are there?
Is this a vegetarian pizza?

Does it appear to be rainy?
Does this person have 20/20 vision?

What color are her eyes?
What is the mustache made of?

Many to one : Visual Question Answering (VQA)

- Entrée : question (texte) et image
- Sortie : réponse (variable qualitative ou quantitative)
- Tâche complexe qui nécessite : compréhension de scène, raisonnement spatial, compréhension de texte.



Bibliographie I


Cho, Kyunghyun et al. (oct. 2014). "Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation". In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2014. Doha, Qatar : Association for Computational Linguistics, p. 1724-1734. DOI : 10.3115/v1/D14-1179. URL : https://aclanthology.org/D14-1179 (visité le 25/05/2023).


Devlin, Jacob et al. (juin 2019). "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding". In : *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota : Association for Computational Linguistics, p. 4171-4186. DOI : 10.18653/v1/N19-1423. URL : https://aclanthology.org/N19-1423.


Hochreiter, Sepp et Jürgen Schmidhuber (nov. 1997). "Long Short-Term Memory". In : *Neural Computation* 9.8, p. 1735-1780. ISSN : 0899-7667. DOI : 10.1162/neco.1997.9.8.1735.


Mikolov, Tomas et al. (16 jan. 2013). "Efficient Estimation of Word Representations in Vector Space". In : ICLR 2013. arXiv : 1301.3781. URL : http://arxiv.org/abs/1301.3781 (visité le 26/03/2019).

Bibliographie II

 Pennington, Jeffrey, Richard Socher et Christopher Manning (2014). "Glove : Global Vectors for Word Representation". In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar). Association for Computational Linguistics, p. 1532-1543. DOI : 10.3115/v1/D14-1162. URL : http://aclweb.org/anthology/D14-1162 (visité le 26/03/2019).

 Shi, Xingjian et al. (7 déc. 2015). "Convolutional LSTM Network : A Machine Learning Approach for Precipitation Nowcasting". In : *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Cambridge, MA, USA : MIT Press, p. 802-810.