

Réseaux de neurones récurrents

Guillaume Bourmaud

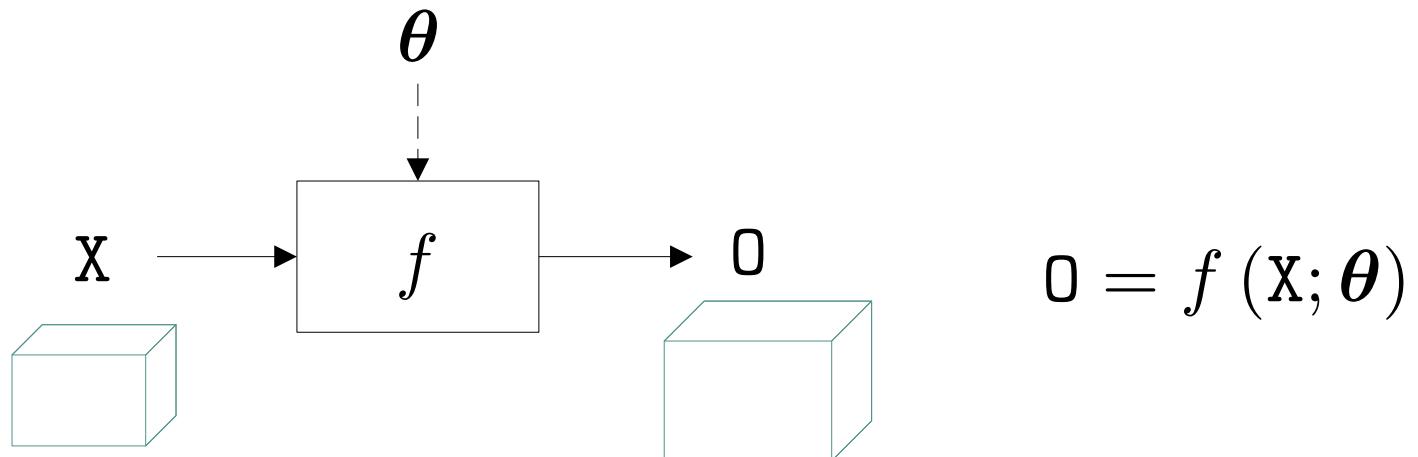
I) Introduction

I)

Rappels : Réseaux de neurones « à propagation avant » pour l'apprentissage supervisé

En anglais « Feed-forward Neural Network » (FNN)

Inférence

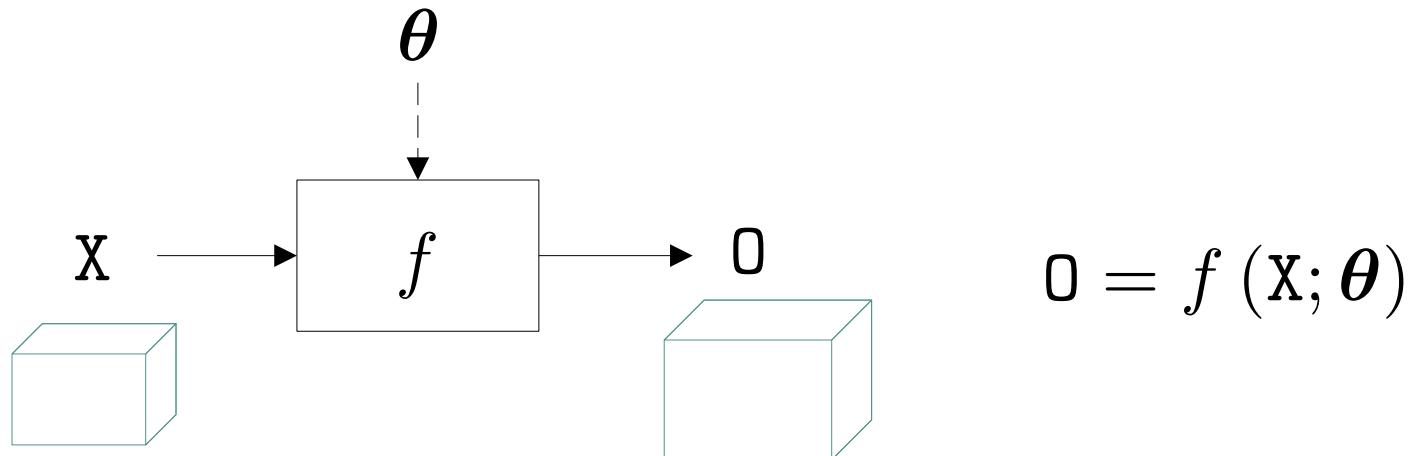


I)

Rappels : Réseaux de neurones « à propagation avant » pour l'apprentissage supervisé

En anglais « Feed-forward Neural Network » (FNN)

Inférence



Apprentissage

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N l \left(Y^{(i)}, f \left(X^{(i)}; \theta \right) \right)$$

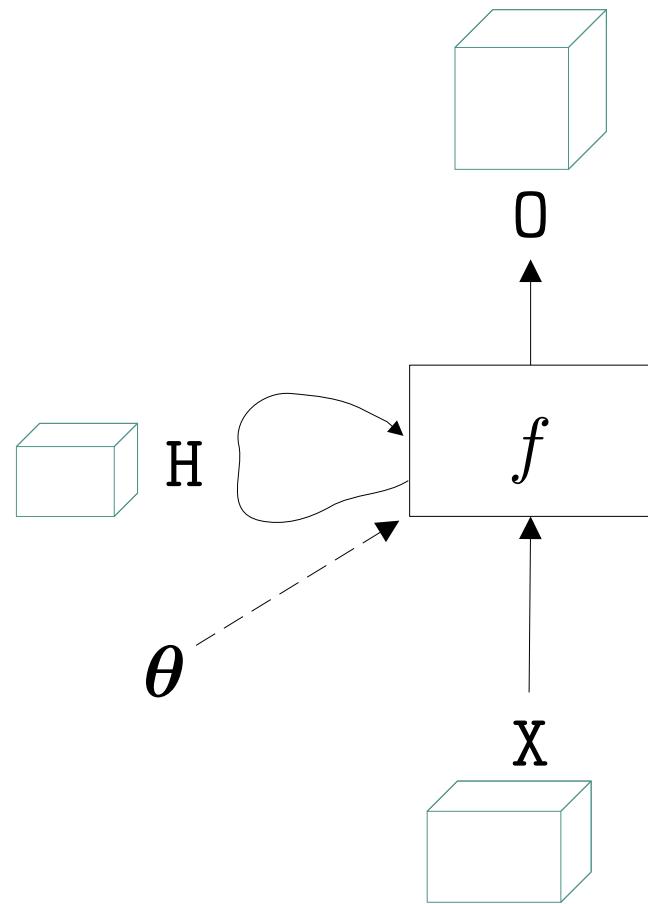
Réseaux de neurones récurrents

- En anglais « Recurrent Neural Network » (RNN)
- Permettent historiquement de gérer facilement les cas où l'entrée et/ou la sortie du réseau ont une taille variable.

```
"the cat sat on the mat" -> [Seq2Seq model] -> "le chat etait assis sur le tapis"
```

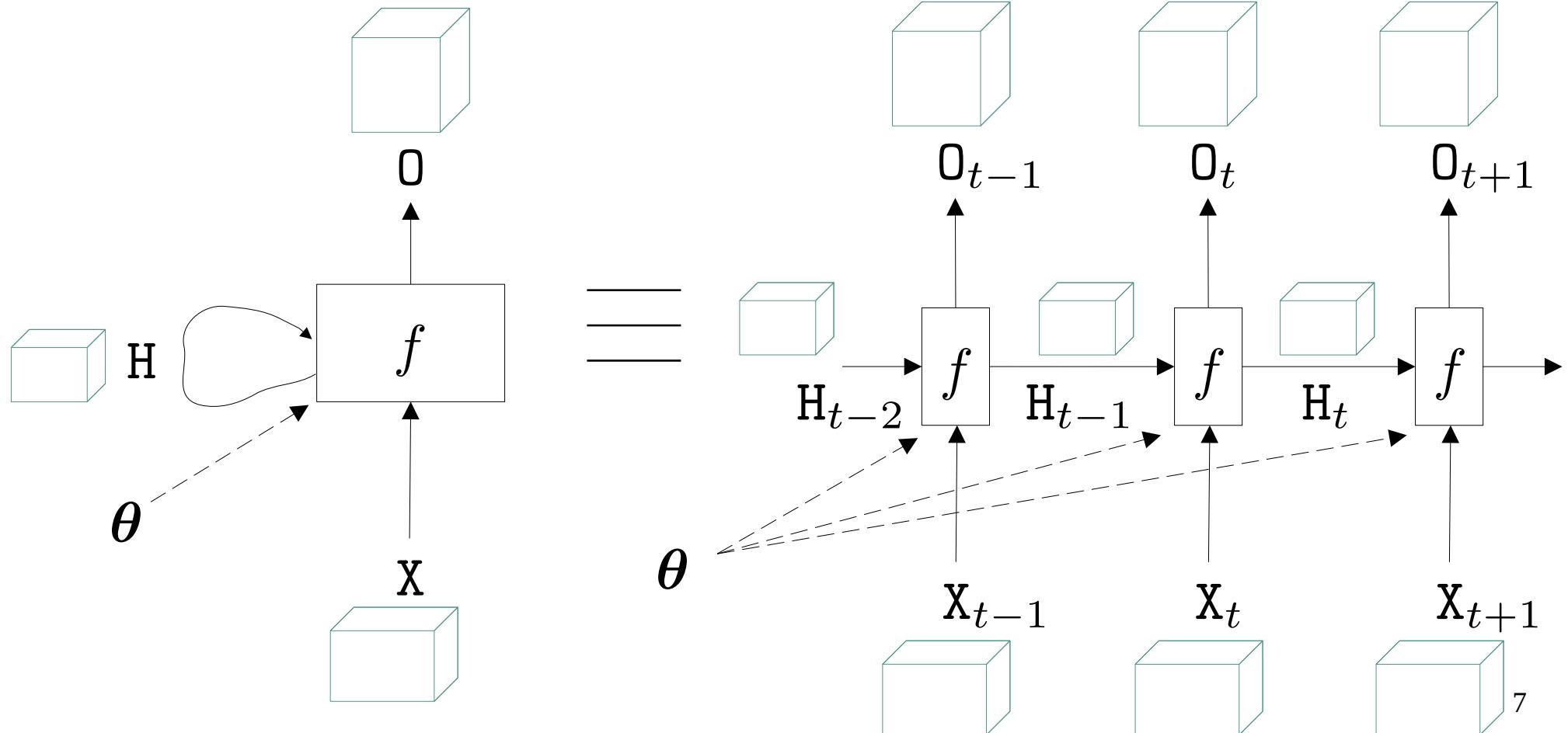
I)

Définition et dépliement d'un réseau de neurones récurrents



I)

Définition et dépliement d'un réseau de neurones récurrents



I)

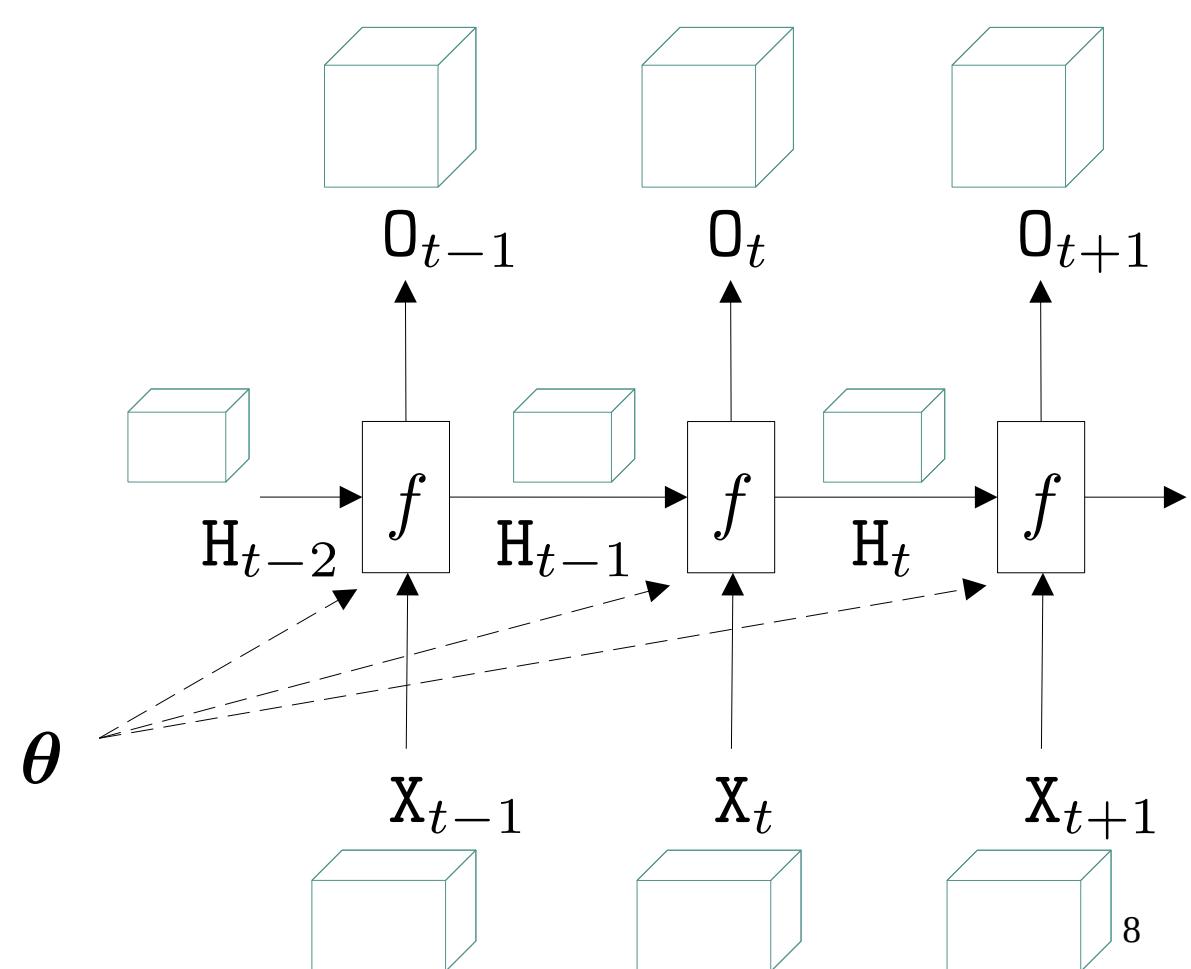
Définition et dépliement d'un réseau de neurones récurrents

Mêmes fonction et paramètres à chaque pas de temps

$$(O_t, H_t) = f(H_{t-1}, X_t; \theta)$$

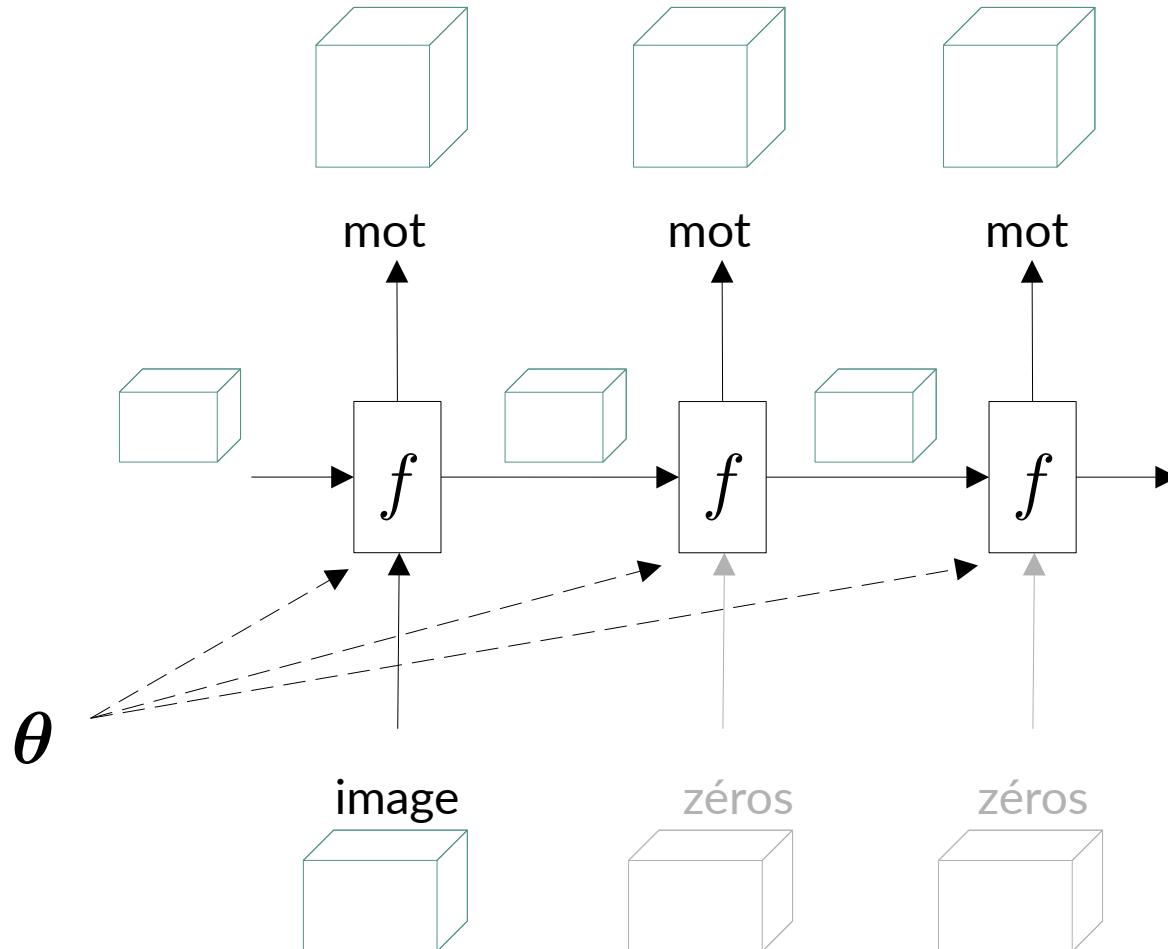
↓
 état caché (joue le rôle de mémoire)
 ↑
 entrée
 ↓
 sortie

Longue dépendance : La sortie à l'instant t dépend de **toutes** les entrées passées.



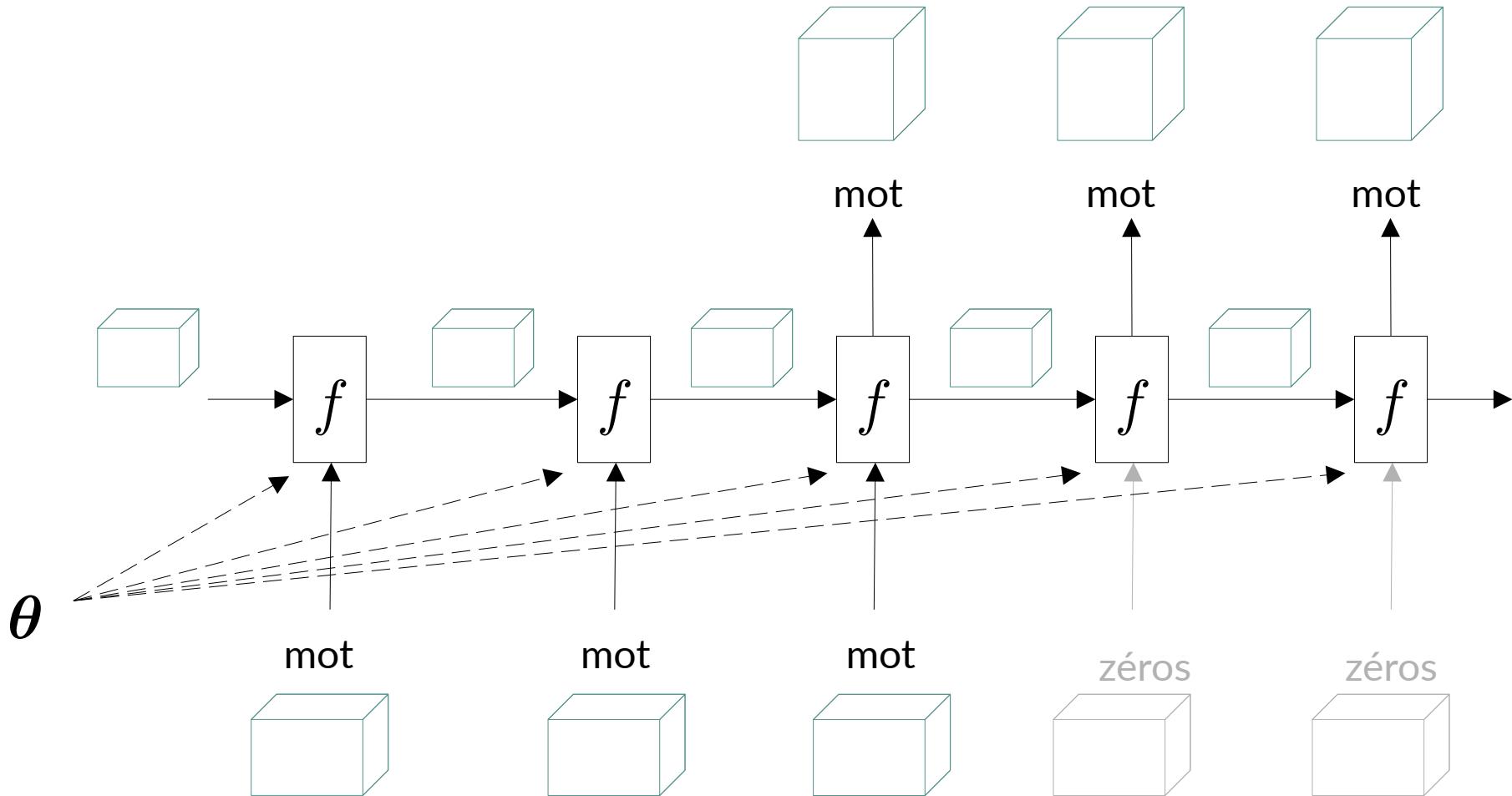
I)

Exemple d'utilisation d'un RNN : description d'une image



I)

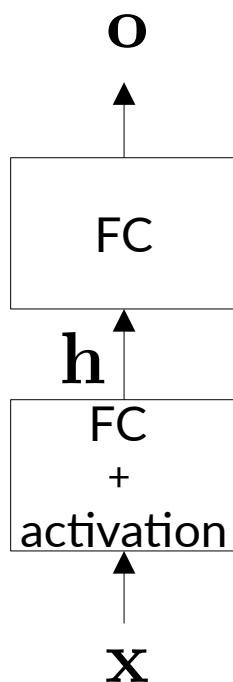
Exemple d'utilisation d'un RNN : traduction de texte



II) Du MLP à une couche cachée au RNN « standard »

II)

Du MLP à une couche cachée au RNN « standard »

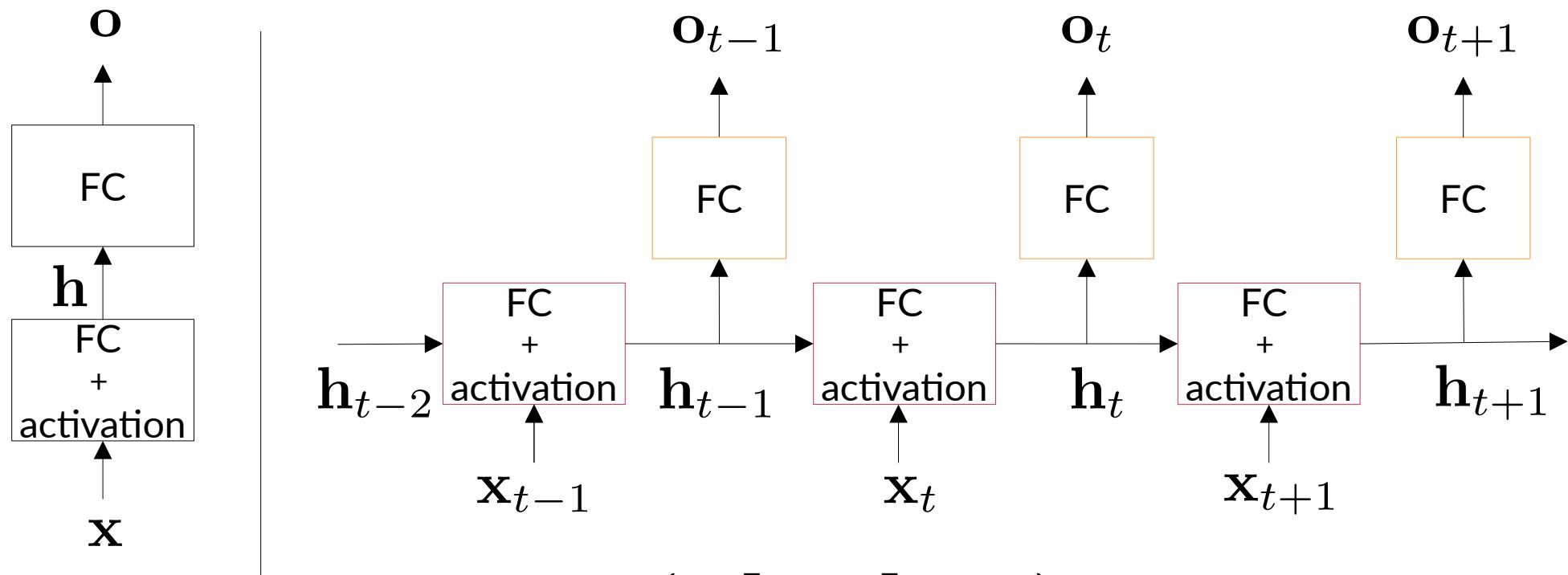


$$\mathbf{h} = \sigma_h (\mathbf{W}_h \mathbf{x} + \mathbf{b}_h)$$

$$\mathbf{o} = \mathbf{W}_o \mathbf{h} + \mathbf{b}_o$$

II)

Du MLP à une couche cachée au RNN « standard »

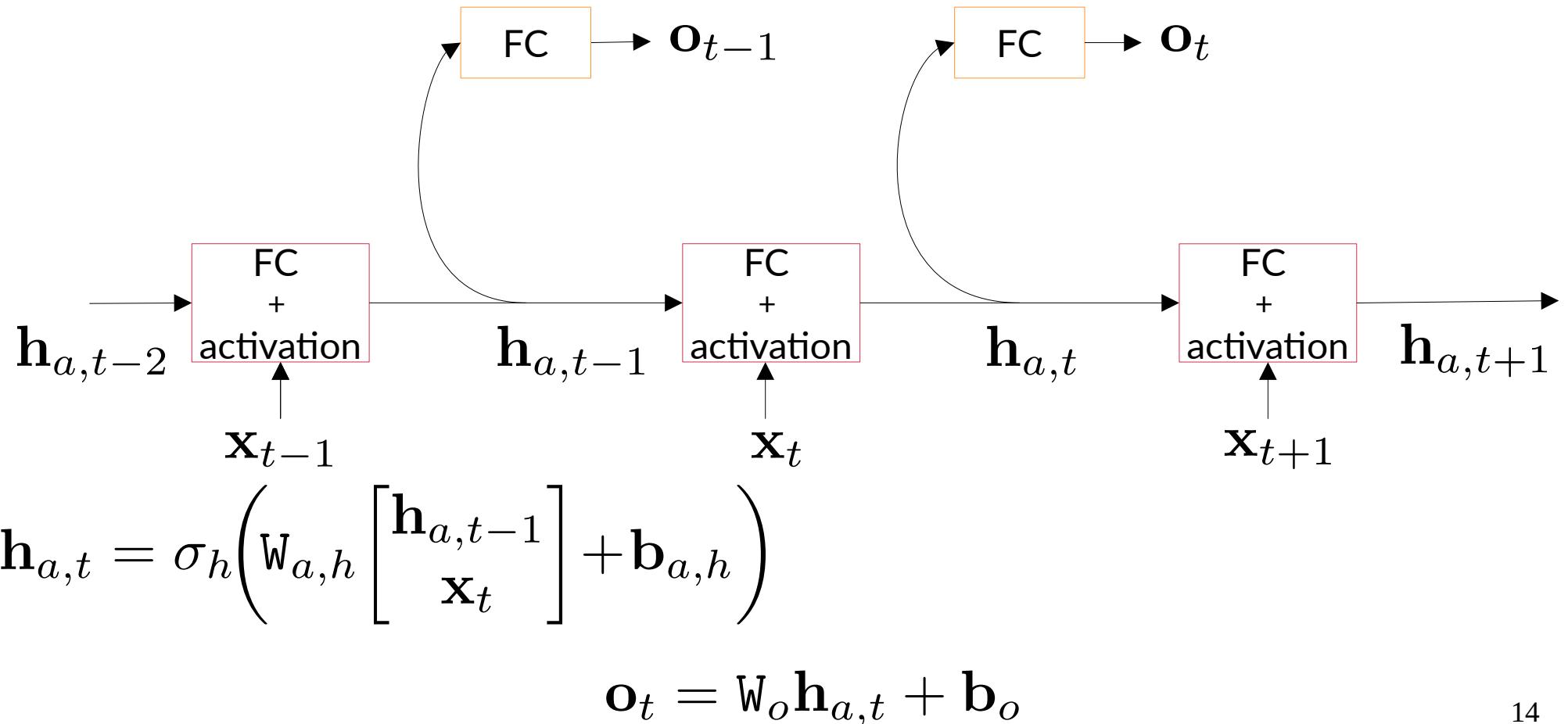


$$\mathbf{h}_t = \sigma_h \left(\mathbf{W}_h \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \right) := f_h (\mathbf{h}_{t-1}, \mathbf{x}_t; \theta_h)$$

$$\mathbf{o}_t = \mathbf{W}_o \mathbf{h}_t + \mathbf{b}_o := f_o (\mathbf{h}_t; \theta_o)$$

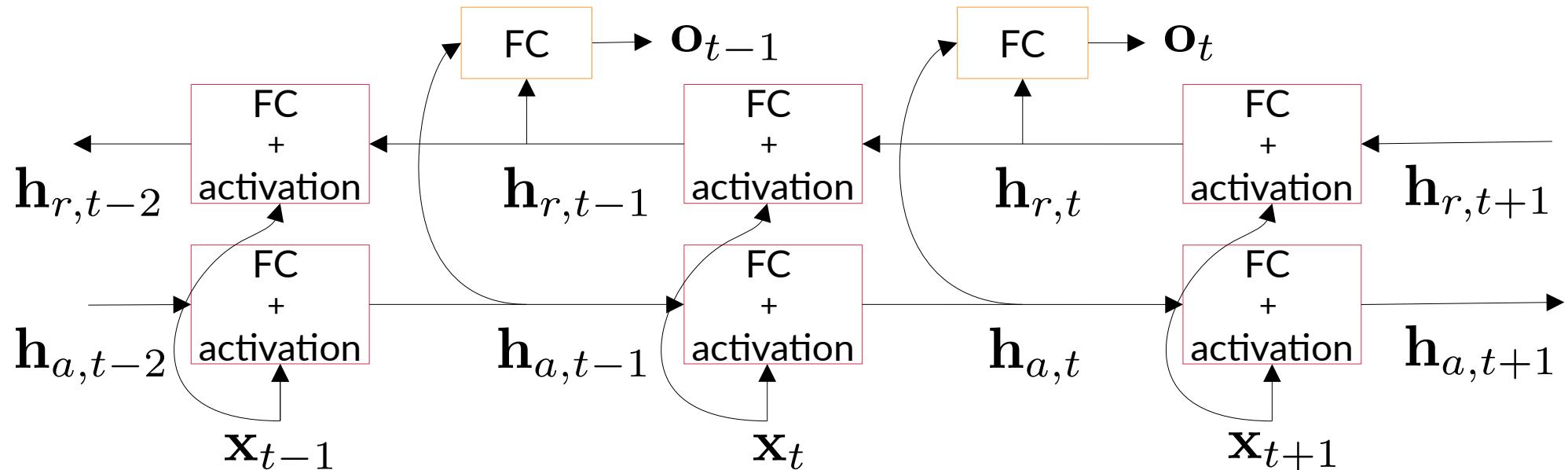
II)

RNN « standard » bidirectionnel



III)

RNN « standard » bidirectionnel



$$\mathbf{h}_{a,t} = \sigma_h \left(\mathbf{W}_{a,h} \begin{bmatrix} \mathbf{h}_{a,t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{a,h} \right)$$

$$\mathbf{h}_{r,t} = \sigma_h \left(\mathbf{W}_{r,h} \begin{bmatrix} \mathbf{h}_{r,t+1} \\ \mathbf{x}_{t+1} \end{bmatrix} + \mathbf{b}_{r,h} \right)$$

$$\mathbf{o}_t = \mathbf{W}_o \begin{bmatrix} \mathbf{h}_{a,t} \\ \mathbf{h}_{r,t} \end{bmatrix} + \mathbf{b}_o$$

II)

Exemple jouet : génération du mot « bille » avec un RNN

Vocabulaire de 4 lettres : i =
 (« one hot encoding »)

$$\begin{array}{c} \text{i} \\ \text{b} \\ \text{e} \\ \text{l} \end{array} = \begin{array}{c} \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \end{matrix} \end{array}$$

Paramètres fixés : $\theta_h = (\mathbf{W}_h, \mathbf{b}_h)$ $\theta_o = (\mathbf{W}_o, \mathbf{b}_o)$

Dimension de l'espace caché : 3

$$\mathbf{h}_0 = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \quad \mathbf{x}_1 = \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix}$$

II)

Exemple jouet : génération du mot « bille » avec un RNN

Vocabulaire de 4 lettres : i =
 (« one hot encoding »)

$$\begin{array}{c} \text{i} \\ \text{b} \\ \text{e} \\ \text{l} \end{array} = \begin{array}{c} \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \end{matrix} \end{array}$$

Paramètres fixés : $\theta_h = (W_h, b_h)$ $\theta_o = (W_o, b_o)$

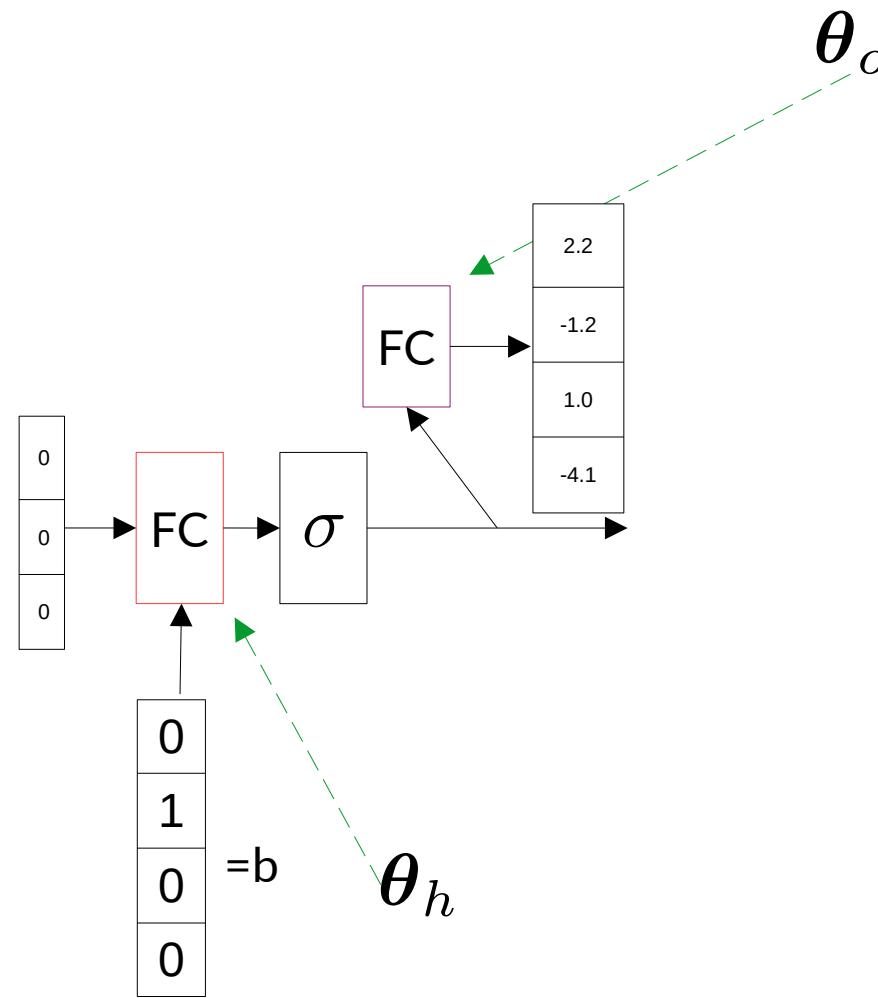
Dimension de l'espace caché : 3

$$\begin{array}{c} \text{h}_0 = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \text{x}_1 = \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix} \end{array}$$

- 1) $\mathbf{h}_t = \sigma_h \left(W_h \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \right)$
 $\mathbf{o}_t = W_o \mathbf{h}_t + \mathbf{b}_o$
 - 2) $t = t + 1, \mathbf{x}_t = \arg \max \mathbf{o}_{t-1}$
 - 3) Aller à 1)

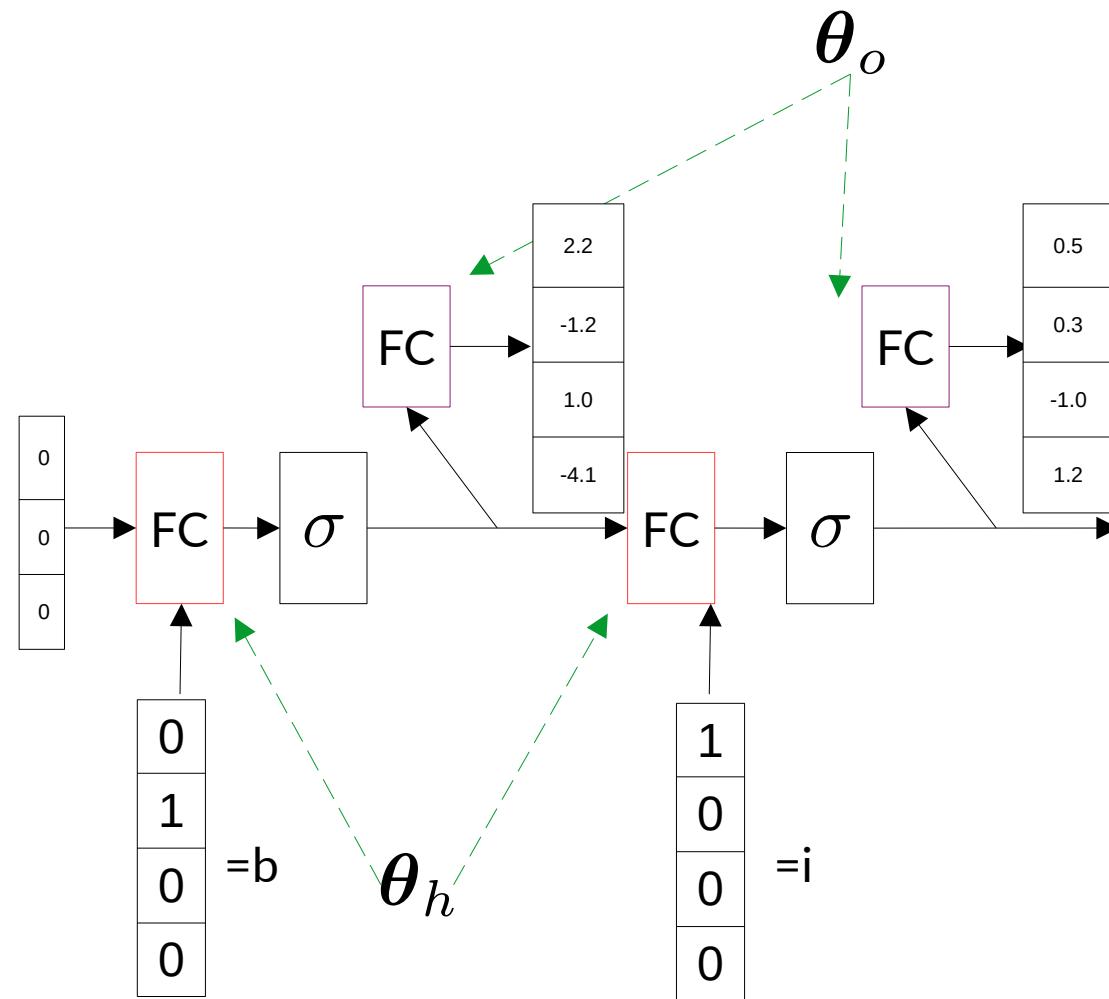
II)

Exemple jouet : génération du mot « bille » avec un RNN (suite)



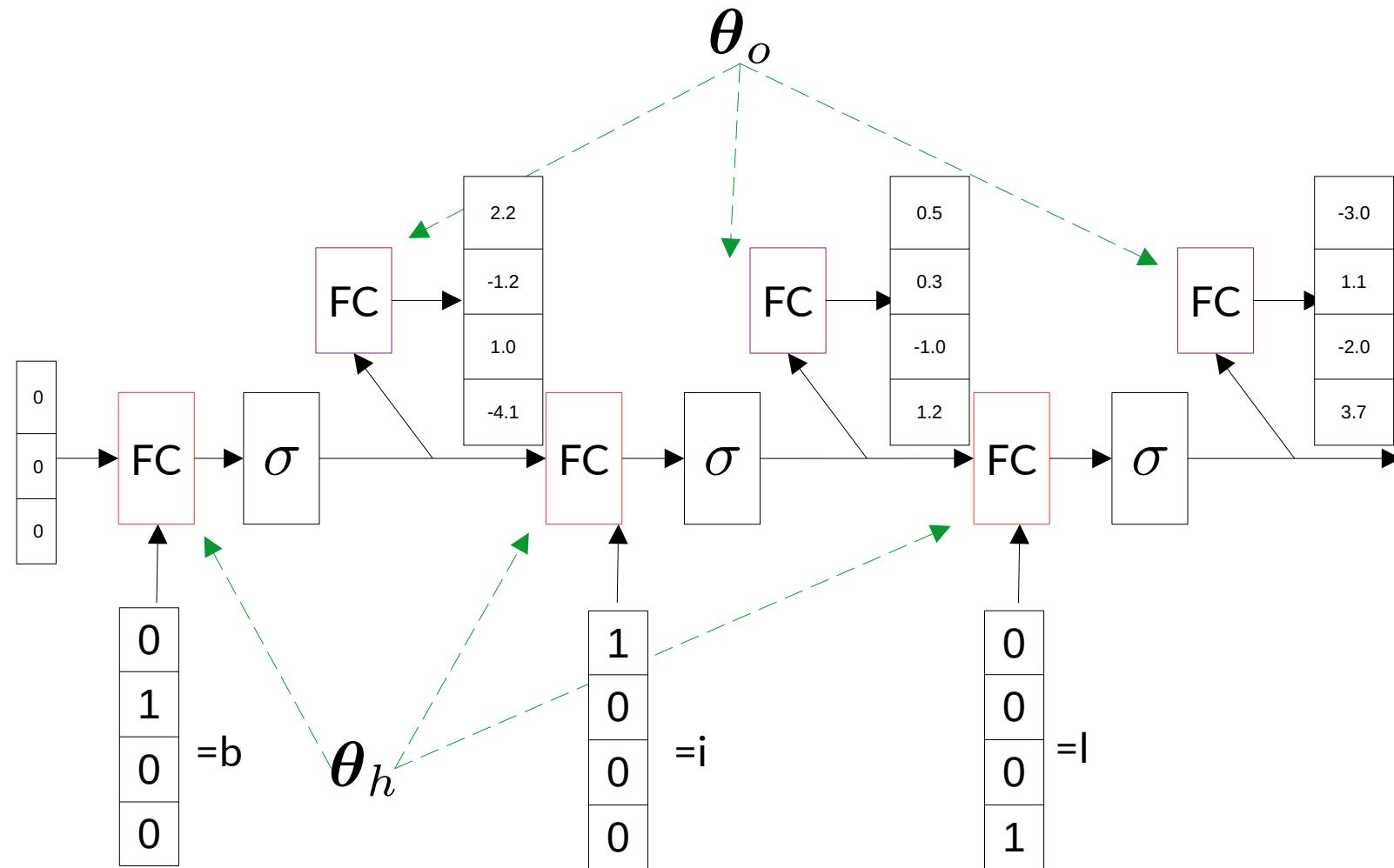
II)

Exemple jouet : génération du mot « bille » avec un RNN (suite)



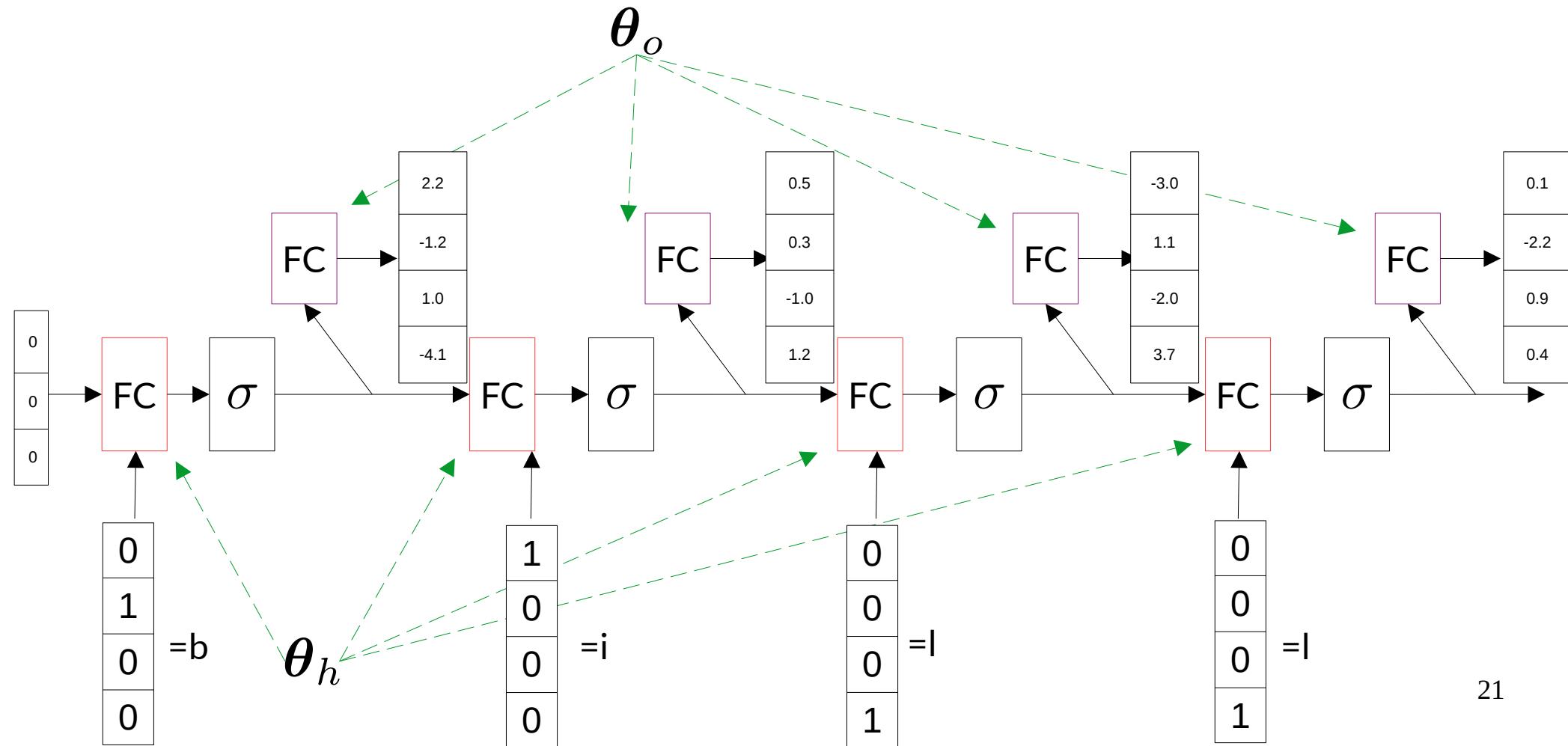
II)

Exemple jouet : génération du mot « bille » avec un RNN (suite)



II)

Exemple jouet : génération du mot « bille » avec un RNN (suite)

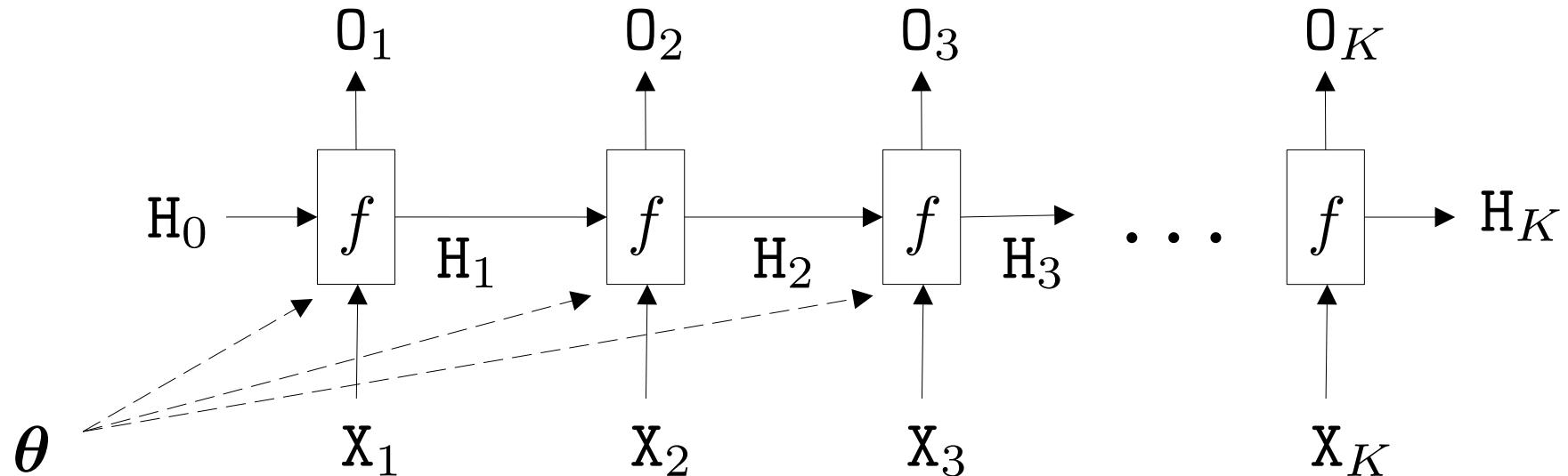


III) Optimisation des paramètres d'un RNN

III)

Optimisation des paramètres d'un RNN

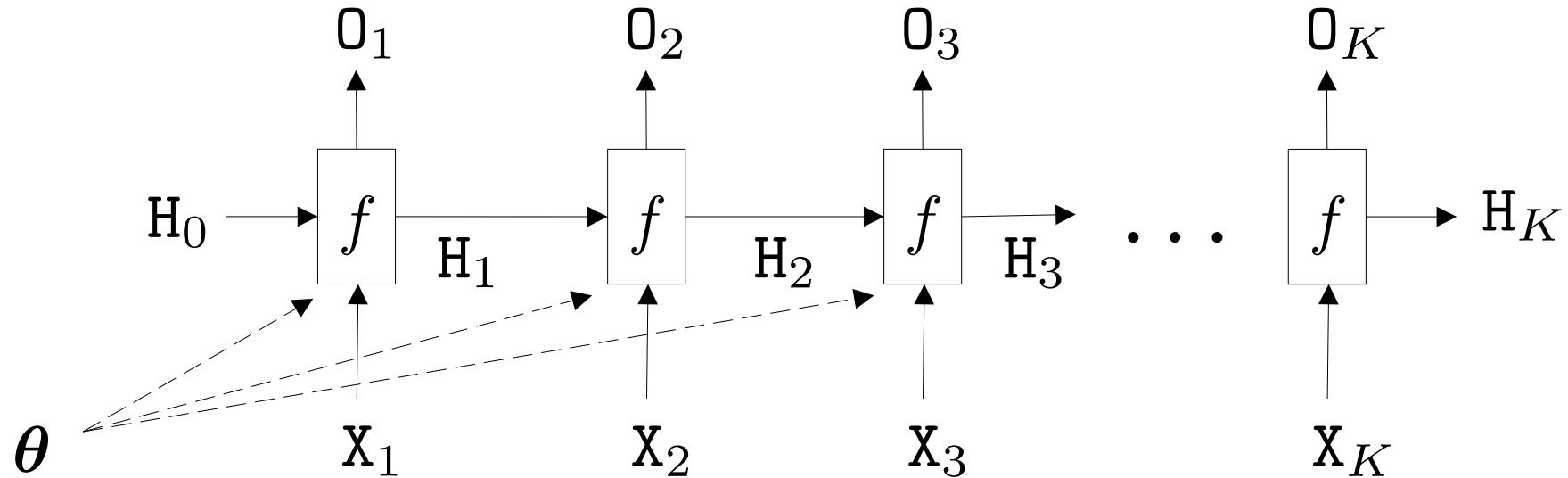
RNN déplié sur K pas de temps = FNN



III)

Optimisation des paramètres d'un RNN

RNN déplié sur K pas de temps = FNN

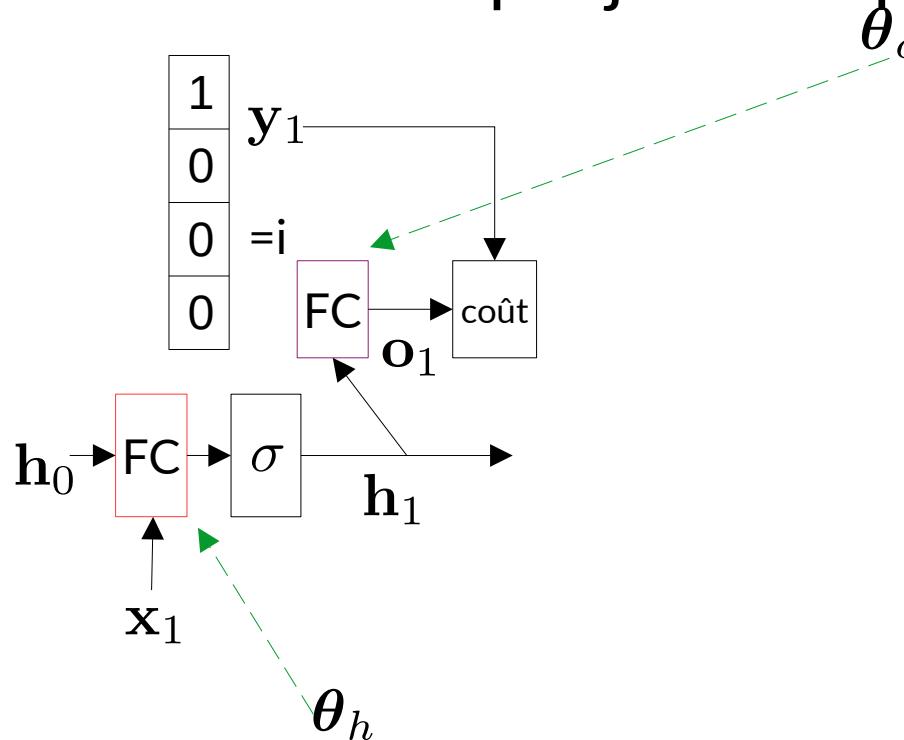


Optimisation des paramètres du RNN = Descente de gradient sur le RNN déplié

En anglais « BackPropagation Through Time » (BPTT)

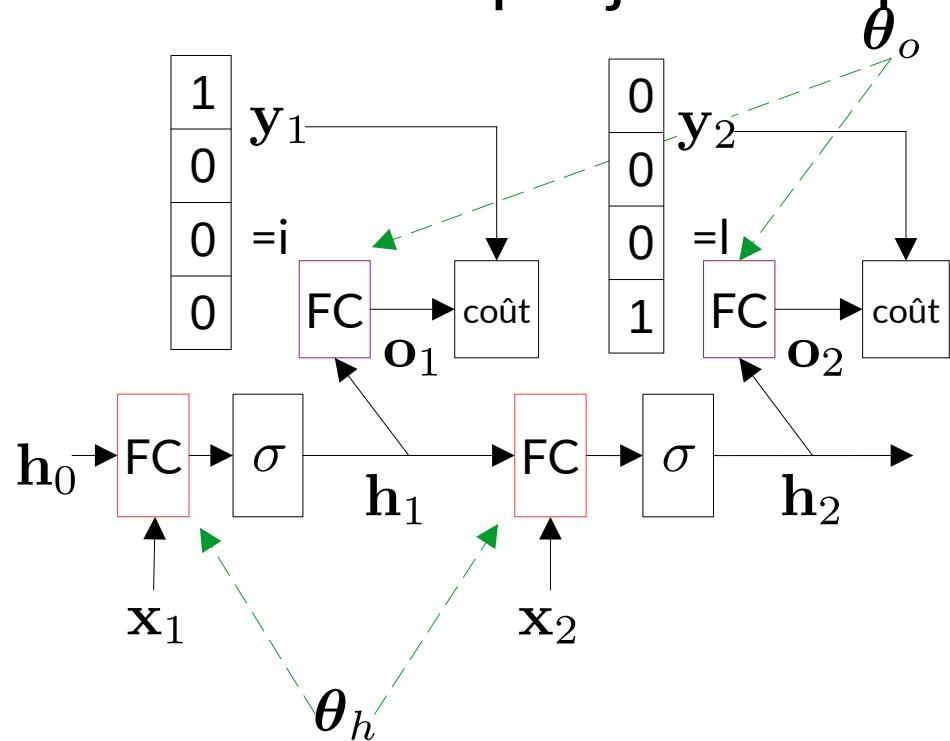
III)

Retour à l'exemple jouet : optimisation des paramètres



$$\theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o))$$

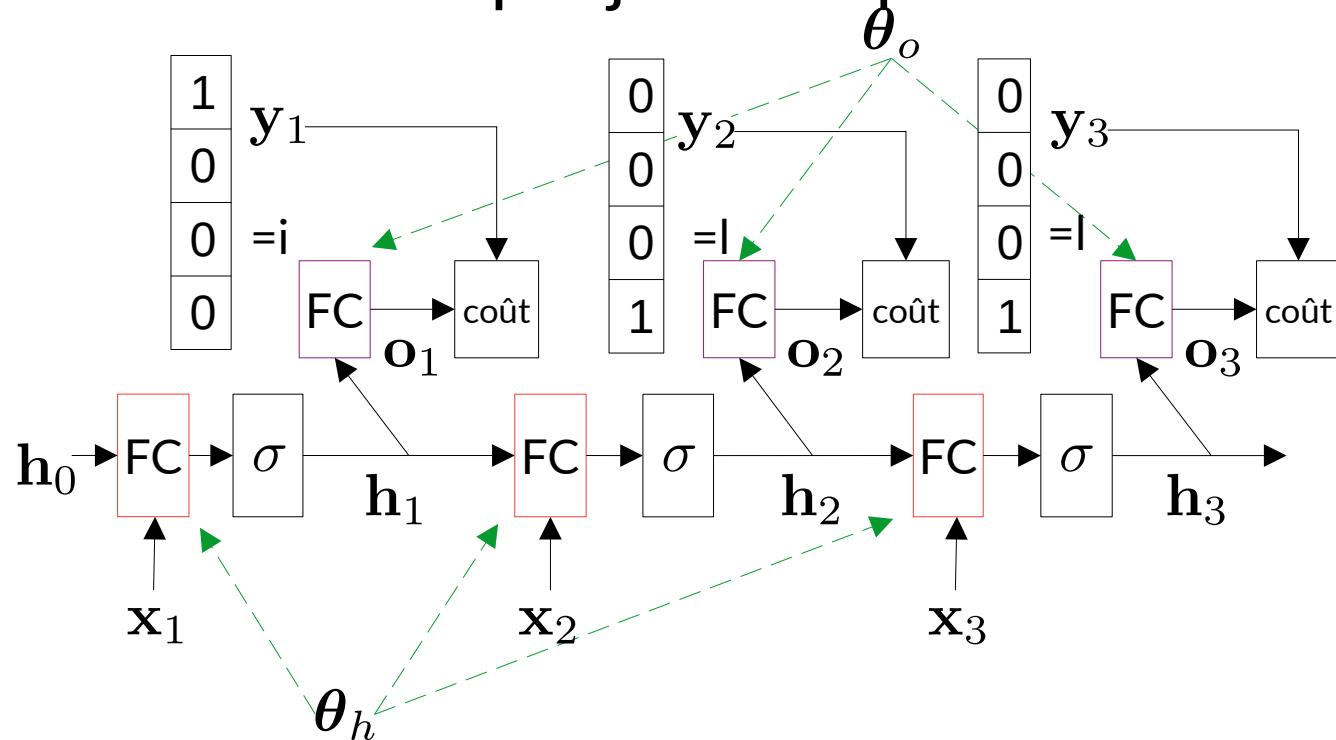
Retour à l'exemple jouet : optimisation des paramètres



$$\theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o))$$

III)

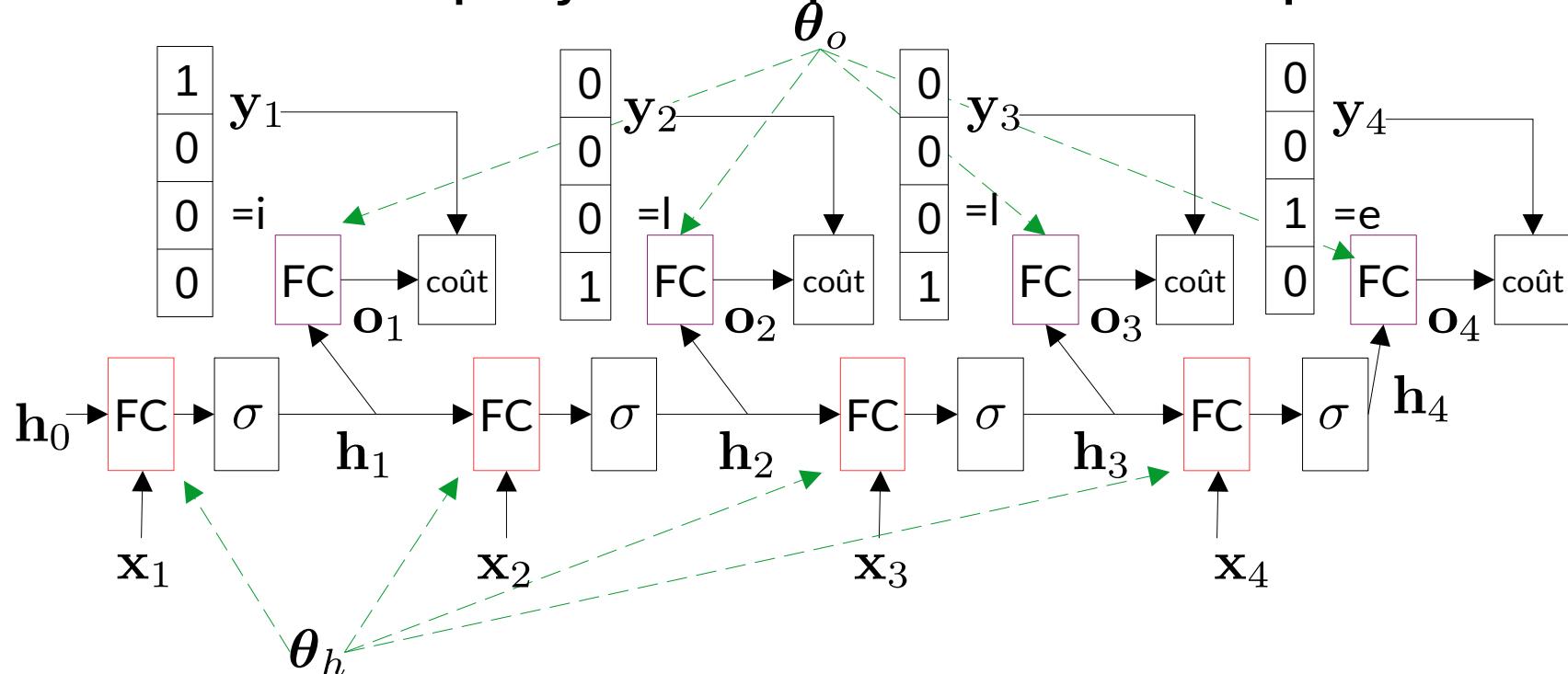
Retour à l'exemple jouet : optimisation des paramètres



$$\theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} \begin{aligned} & l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \end{aligned}$$

III)

Retour à l'exemple jouet : optimisation des paramètres



$$\theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} \begin{aligned} & l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_4, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o)) \end{aligned}$$

III)

Retour à l'exemple jouet : optimisation des paramètres

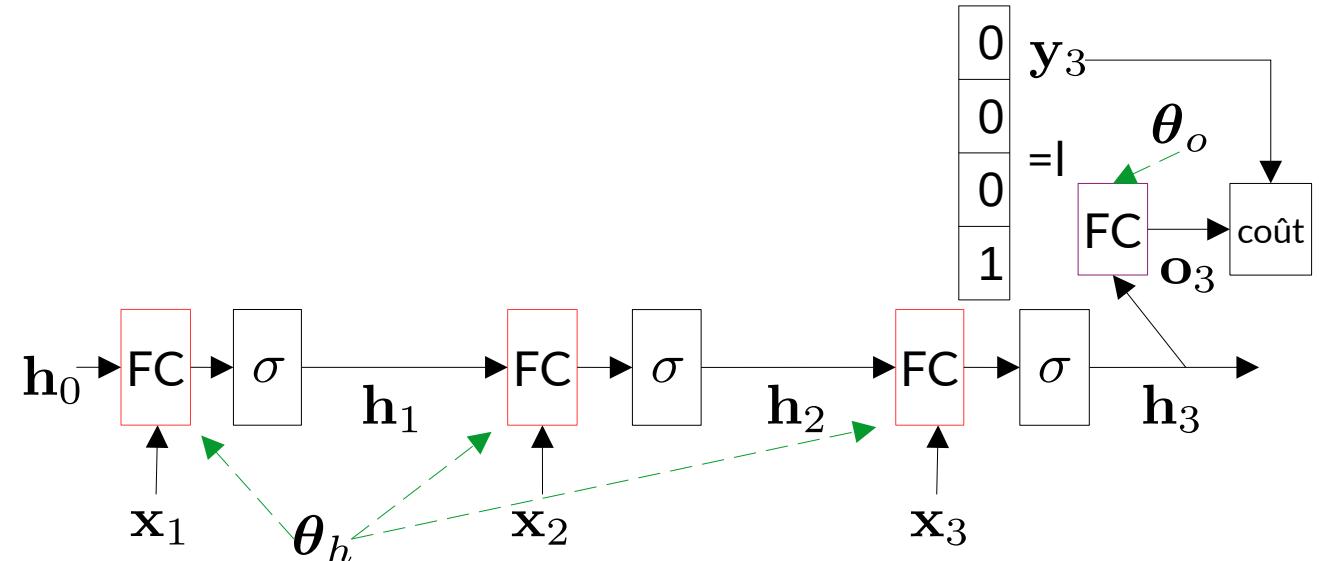
$$\begin{aligned}
 \theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} & l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\
 & + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\
 & + l(\mathbf{y}_3, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \\
 & + l(\mathbf{y}_4, f_o(f_h(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o))
 \end{aligned}$$

30

III)

Retour à l'exemple jouet : optimisation des paramètres

$$\frac{\partial}{\partial \theta_h} f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h) =$$



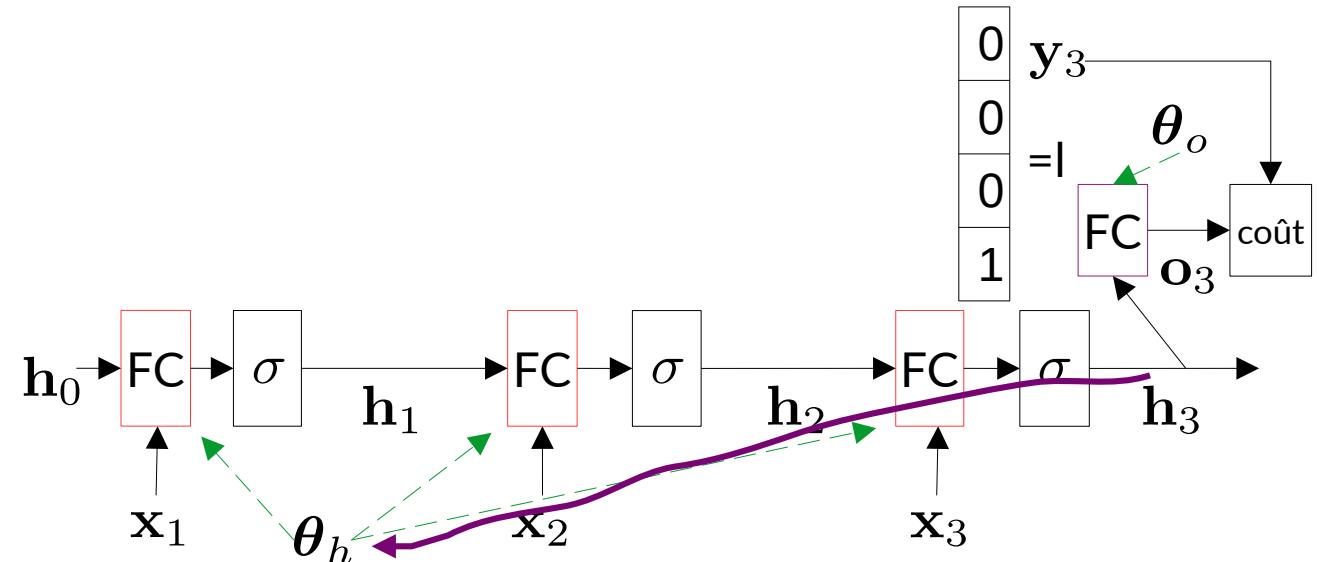
$$\begin{aligned} \theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} & l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_4, f_o(f_h(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o)) \end{aligned}$$

III)

Retour à l'exemple jouet : optimisation des paramètres

$$\frac{\partial}{\partial \theta_h} f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h) =$$

$$+ \boxed{\frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h)}$$



$$\begin{aligned} \theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} & l(\mathbf{y}_1, f_o(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \quad 32 \\ & + l(\mathbf{y}_4, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o)) \end{aligned}$$

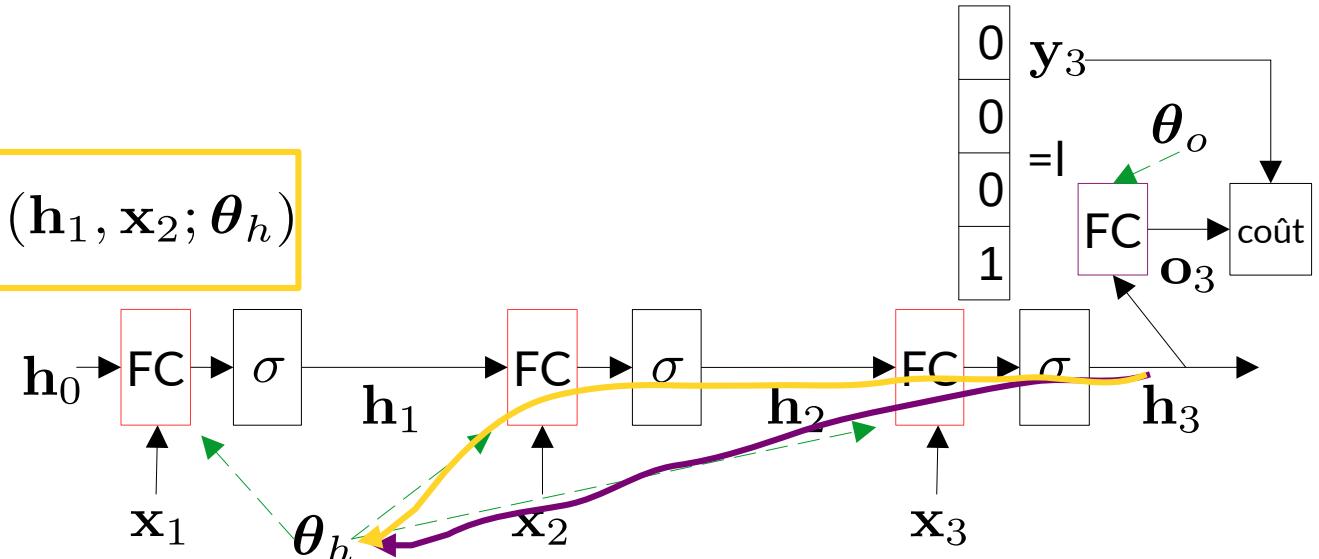
III)

Retour à l'exemple jouet : optimisation des paramètres

$$\frac{\partial}{\partial \theta_h} f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h) =$$

$$+ \boxed{\frac{\partial}{\partial \mathbf{h}_2} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h) \frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_1, \mathbf{x}_2; \theta_h)}$$

$$+ \boxed{\frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h)}$$



$$\begin{aligned} \theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} & l(\mathbf{y}_1, f_o(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \quad 33 \\ & + l(\mathbf{y}_4, f_o(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{f}_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o)) \end{aligned}$$

III)

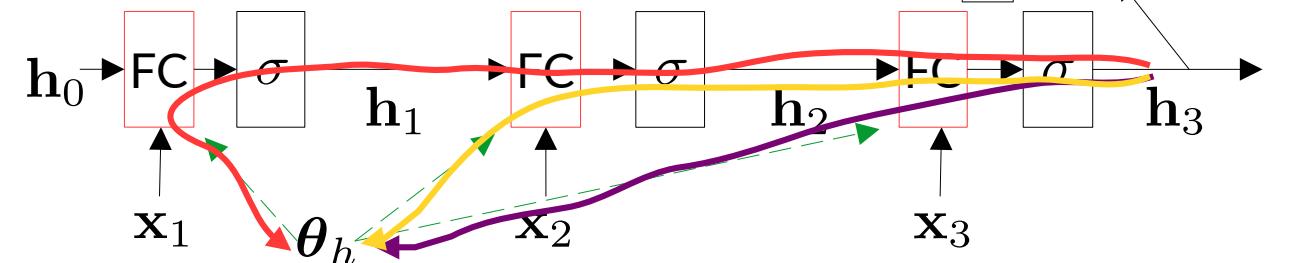
Retour à l'exemple jouet : optimisation des paramètres

$$\frac{\partial}{\partial \theta_h} f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h) =$$

$$\frac{\partial}{\partial \mathbf{h}_2} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h) \cdot \frac{\partial}{\partial \mathbf{h}_1} f_h(\mathbf{h}_1, \mathbf{x}_2; \theta_h) \cdot \frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h)$$

$$+ \frac{\partial}{\partial \mathbf{h}_2} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h) \frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_1, \mathbf{x}_2; \theta_h)$$

$$+ \frac{\partial}{\partial \theta_h} f_h(\mathbf{h}_2, \mathbf{x}_3; \theta_h)$$

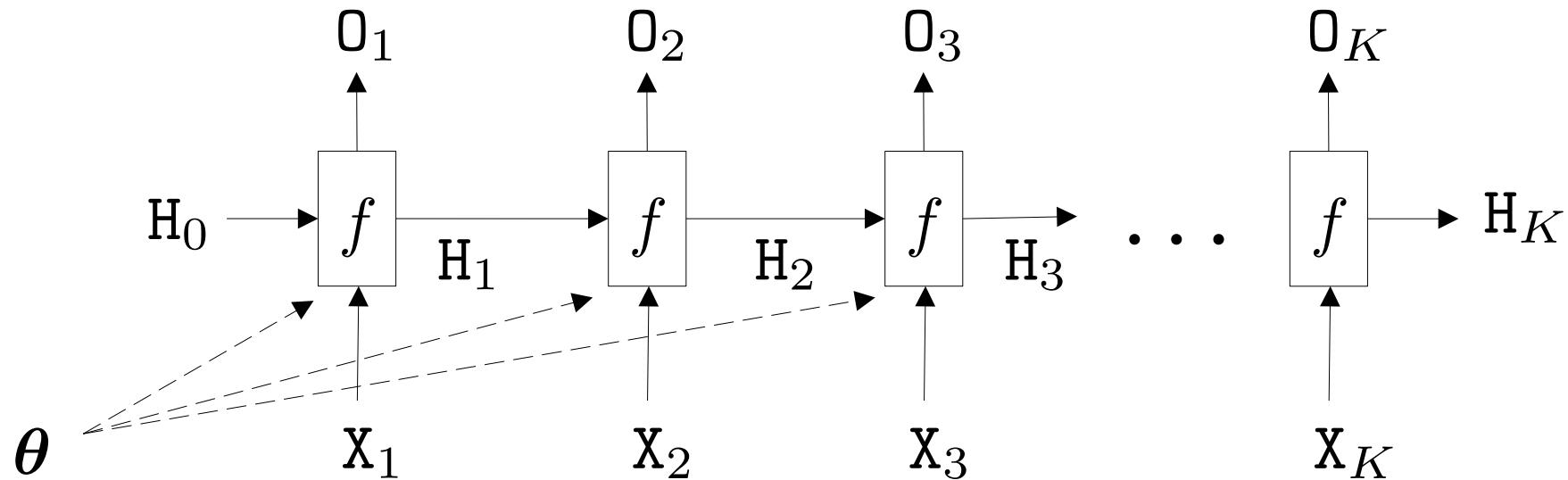


$$\begin{aligned} \theta_o^*, \theta_h^* = \arg \min_{\theta_o, \theta_h} & l(\mathbf{y}_1, f_o(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_2, f_o(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_3, f_o(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h); \theta_o)) \\ & + l(\mathbf{y}_4, f_o(f_h(f_h(f_h(f_h(\mathbf{h}_0, \mathbf{x}_1; \theta_h), \mathbf{x}_2; \theta_h), \mathbf{x}_3; \theta_h), \mathbf{x}_4; \theta_h); \theta_o)) \end{aligned}$$

IV) Le problème de l'apprentissage de longues dépendances

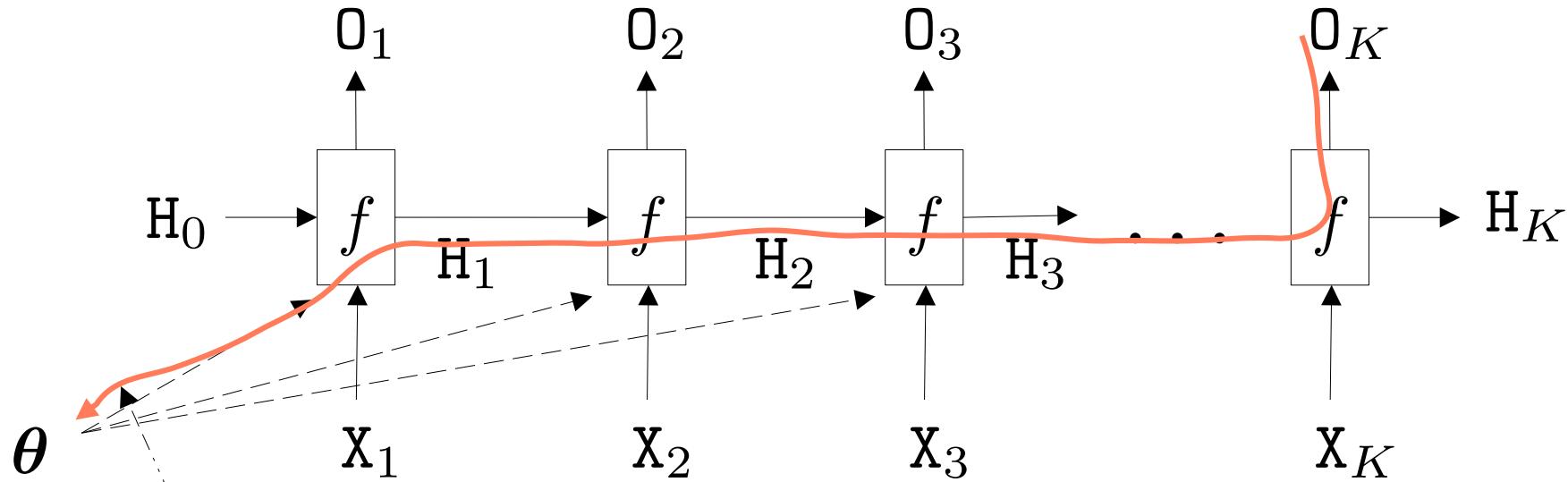
Le problème de l'apprentissage de longues dépendances

Longue dépendance = la sortie à un instant dépend de toutes les entrées précédentes



Le problème de l'apprentissage de longues dépendances

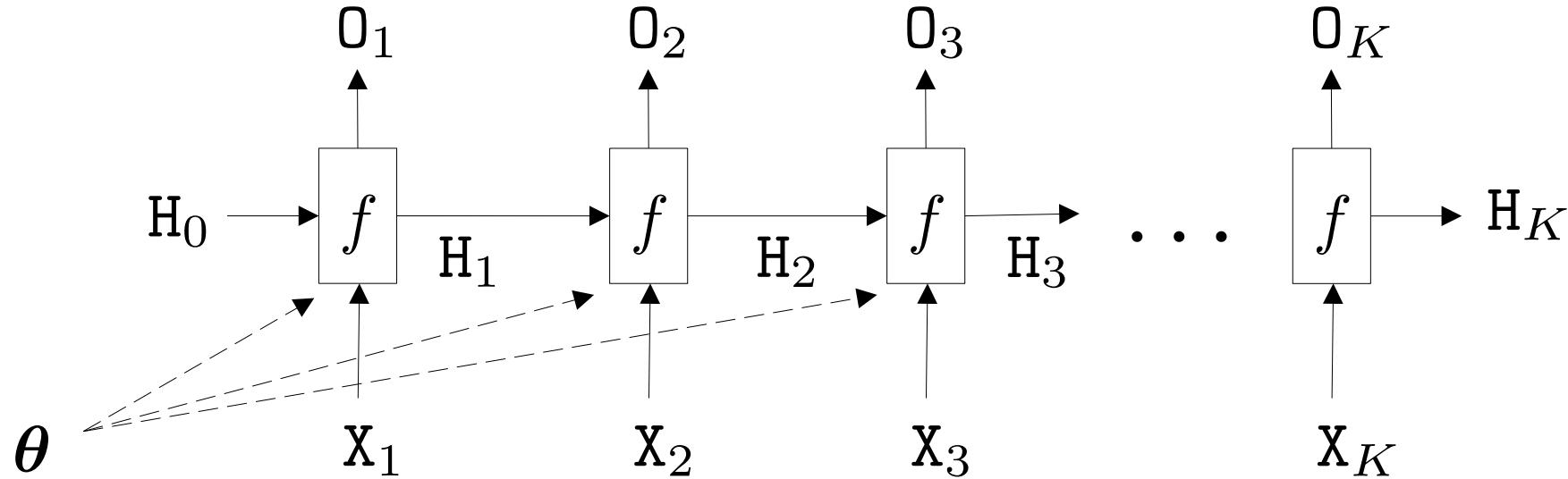
Longue dépendance = la sortie à un instant dépend de toutes les entrées précédentes



Pour que le réseau puisse apprendre les « corrélations » entre O_K et X_1 , il faut notamment que ce gradient soit stable et non nul.

Le problème de l'apprentissage de longues dépendances (suite)

RNN déplié sur K pas de temps = FNN à K couches



K grand = FNN profond \rightarrow risque d'évanouissement ou d'explosion du gradient

Si explosion

\rightarrow aucun apprentissage

Si évanouissement \rightarrow apprentissage uniquement de courtes dépendances

Évanouissement/explosion du gradient dans un RNN

RNN « standard » : fonction d'activation = tanh

Problème : Gradient de tanh entre 0 et 1

→ multiplication du gradient par un facteur positif inférieur à 1 lors de la rétropropagation = affaiblissement systématique du gradient à chaque pas de temps

Évanouissement/explosion du gradient dans un RNN

RNN « standard » : fonction d'activation = tanh

Problème : Gradient de tanh entre 0 et 1

→ multiplication du gradient par un facteur positif inférieur à 1 lors de la rétropropagation = affaiblissement systématique du gradient à chaque pas de temps

Solution 1 : Remplacer tanh par ReLu

Lors de la rétroproagation, le gradient est multiplié exactement soit par 0 soit par 1

Évanouissement/explosion du gradient dans un RNN

RNN « standard » : fonction d'activation = tanh

Problème : Gradient de tanh entre 0 et 1

→ multiplication du gradient par un facteur positif inférieur à 1 lors de la rétropropagation = affaiblissement systématique du gradient à chaque pas de temps

Solution 1 : Remplacer tanh par ReLu

Lors de la rétroproagation, le gradient est multiplié exactement soit par 0 soit par 1

Solution 2 : Changement d'architecture → LSTM / GRU

Évanouissement/explosion du gradient dans un RNN (suite)

En supposant que la fonction d'activation soit l'identité, on a :

$$\mathbf{h}_t = \begin{bmatrix} \mathbf{w}_{hh} & \mathbf{w}_{hx} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \quad \frac{\partial}{\partial \mathbf{h}_{t-1}} \mathbf{h}_t = \mathbf{w}_{hh}$$

$$\frac{\partial}{\partial \mathbf{h}_{t-K}} \mathbf{h}_t = \mathbf{w}_{hh}^K \quad \leftarrow \text{Matrice à la puissance K} \rightarrow \text{évanouissement ou explosion du gradient}$$

Évanouissement/explosion du gradient dans un RNN (suite)

En supposant que la fonction d'activation soit l'identité, on a :

$$\mathbf{h}_t = \begin{bmatrix} \mathbf{w}_{hh} & \mathbf{w}_{hx} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \quad \frac{\partial}{\partial \mathbf{h}_{t-1}} \mathbf{h}_t = \mathbf{w}_{hh}$$

$$\frac{\partial}{\partial \mathbf{h}_{t-K}} \mathbf{h}_t = \mathbf{w}_{hh}^K \quad \leftarrow \text{Matrice à la puissance K} \rightarrow \text{évanouissement ou explosion du gradient}$$

Solution 1 : Initialisation $\mathbf{W}_{hh} = \mathbf{I}$ $\mathbf{b}_h = \mathbf{0}$

Variante : Initialisation en « contrôlant » les valeurs propres (« Echo-State Network »)

Évanouissement/explosion du gradient dans un RNN (suite)

En supposant que la fonction d'activation soit l'identité, on a :

$$\mathbf{h}_t = [W_{hh} \ W_{hx}] \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \quad \frac{\partial}{\partial \mathbf{h}_{t-1}} \mathbf{h}_t = W_{hh}$$

$$\frac{\partial}{\partial \mathbf{h}_{t-K}} \mathbf{h}_t = W_{hh}^K \quad \leftarrow \text{Matrice à la puissance K} \rightarrow \text{évanouissement ou explosion du gradient}$$

Solution 1 : Initialisation $W_{hh} = I \quad \mathbf{b}_h = \mathbf{0}$

Variante : Initialisation en « contrôlant » les valeurs propres (« Echo-State Network »)

Solution 2 : Saturation du gradient (« Gradient Clipping »)

Évanouissement/explosion du gradient dans un RNN (suite)

En supposant que la fonction d'activation soit l'identité, on a :

$$\mathbf{h}_t = [W_{hh} \ W_{hx}] \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h \quad \frac{\partial}{\partial \mathbf{h}_{t-1}} \mathbf{h}_t = W_{hh}$$

$$\frac{\partial}{\partial \mathbf{h}_{t-K}} \mathbf{h}_t = W_{hh}^K \quad \leftarrow \text{Matrice à la puissance K} \rightarrow \text{évanouissement ou explosion du gradient}$$

Solution 1 : Initialisation $W_{hh} = I \quad \mathbf{b}_h = \mathbf{0}$

Variante : Initialisation en « contrôlant » les valeurs propres (« Echo-State Network »)

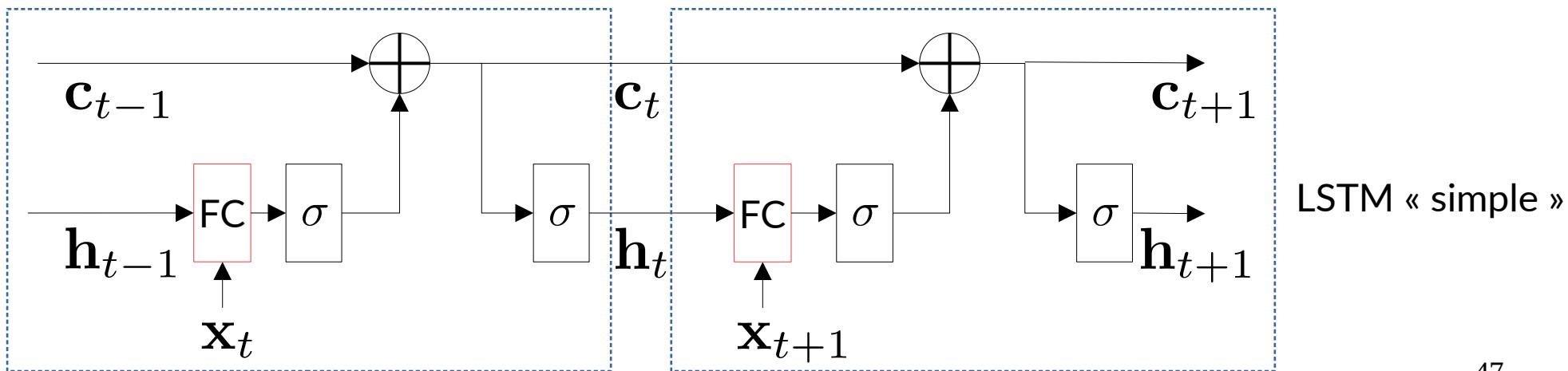
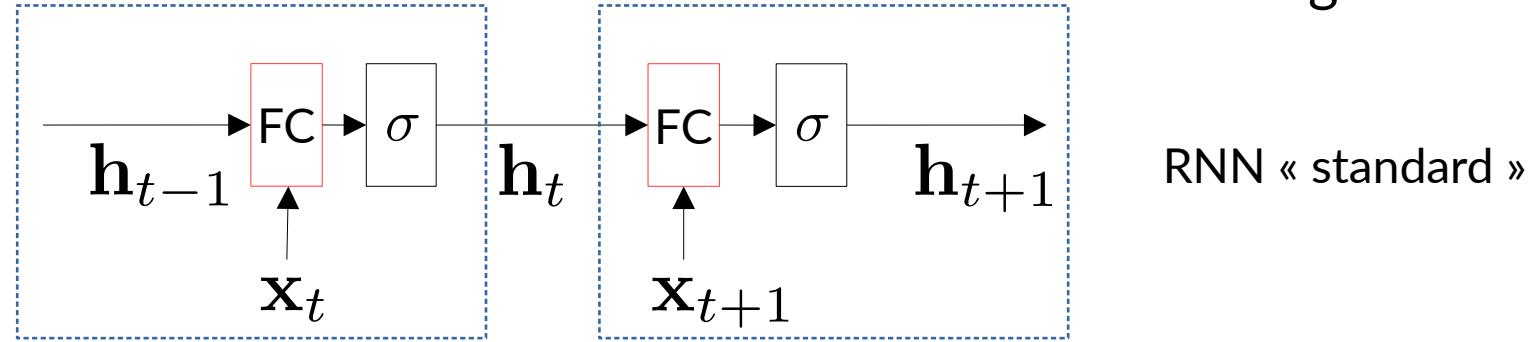
Solution 2 : Saturation du gradient (« Gradient Clipping)

Solution 3 : Changement d'architecture → LSTM / GRU

V) Architectures LSTM / GRU

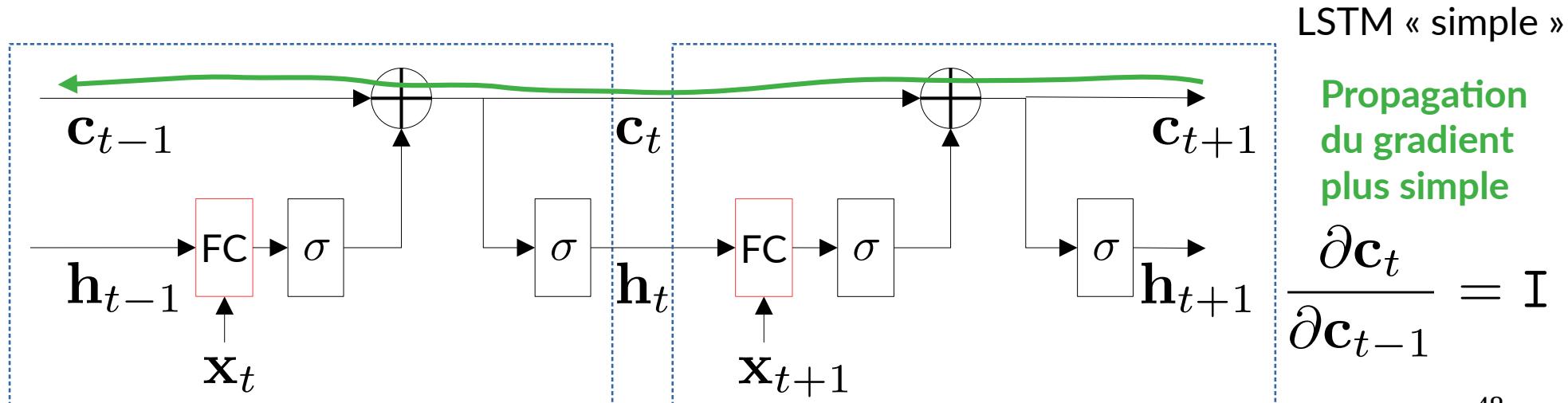
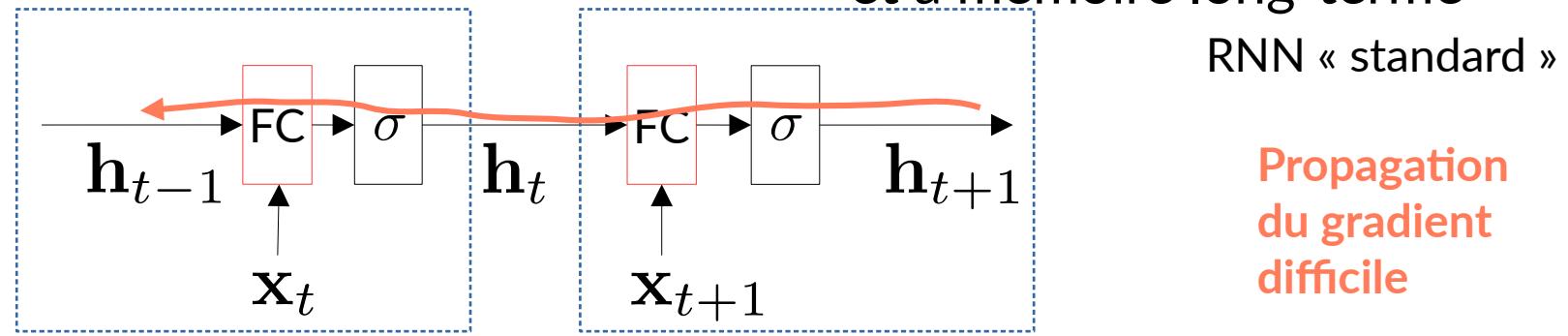
V)

« Long-Short Term Memory » (LSTM) = Réseau à mémoire court-terme et à mémoire long-terme



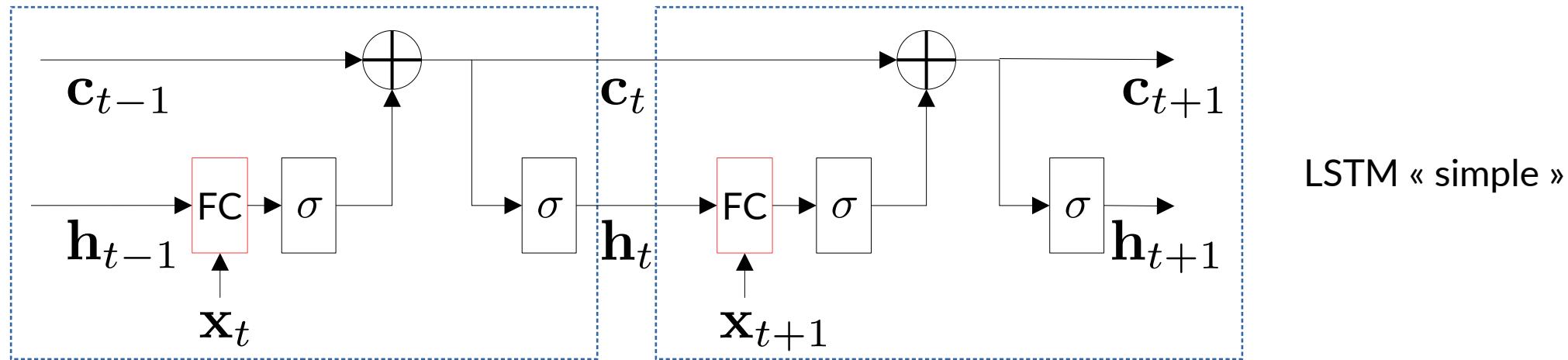
V)

« Long-Short Term Memory » (LSTM) = Réseau à mémoire court-terme et à mémoire long-terme



V)

« Long-Short Term Memory » (LSTM) = Réseau à mémoire court-terme et à mémoire long-terme



Cellule
(joue le rôle
de mémoire)

État caché $\rightarrow \mathbf{h}_t = \tanh(\mathbf{c}_t)$

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \tanh(\mathbf{W}_h \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h)$$

Connexion résiduelle, comme
dans ResNet (2015)
Rappel : LSTM (1998)

V)

Sigmoïde (sortie dans $[0, 1]$)

LSTM « complet »

$$\mathbf{g}_{f_t} = \sigma(\mathbf{W}_{g_f} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_f}) \quad \text{« forget gate » pour effacer la mémoire}$$

$$\mathbf{g}_{i_t} = \sigma(\mathbf{W}_{g_i} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_i}) \quad \text{« input gate » pour écrire dans la mémoire}$$

$$\mathbf{g}_{o_t} = \sigma(\mathbf{W}_{g_o} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_o}) \quad \text{« output gate » pour lire dans la mémoire}$$

$$\mathbf{c}_t = \mathbf{g}_{f_t} \cdot \mathbf{c}_{t-1} + \mathbf{g}_{i_t} \cdot \tanh(\mathbf{W}_h \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h)$$

$$\mathbf{h}_t = \mathbf{g}_{o_t} \cdot \tanh(\mathbf{c}_t)$$

V)

Sigmoïde (sortie dans $[0, 1]$)

LSTM « complet »

$$\mathbf{g}_{f_t} = \sigma(\mathbf{W}_{g_f} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_f}) \quad \text{« forget gate » pour effacer la mémoire}$$

$$\mathbf{g}_{i_t} = \sigma(\mathbf{W}_{g_i} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_i}) \quad \text{« input gate » pour écrire dans la mémoire}$$

$$\mathbf{g}_{o_t} = \sigma(\mathbf{W}_{g_o} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_o}) \quad \text{« output gate » pour lire dans la mémoire}$$

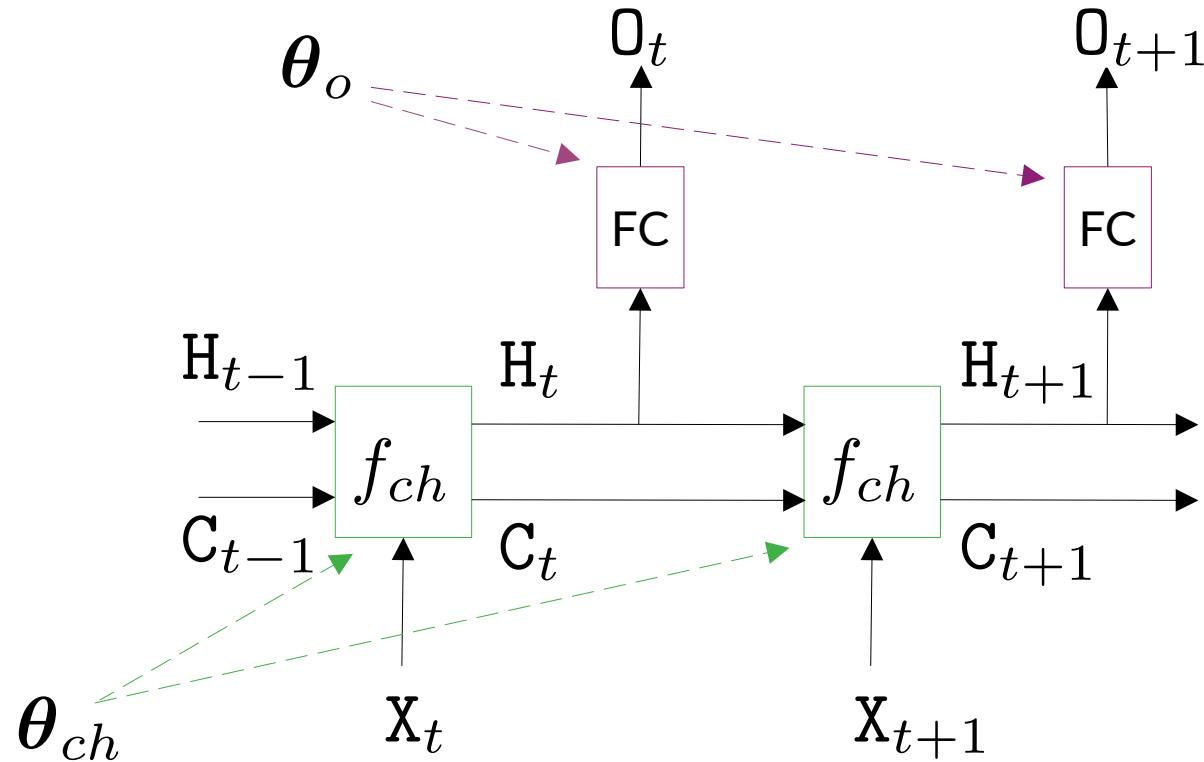
$$\mathbf{c}_t = \mathbf{g}_{f_t} \cdot \mathbf{c}_{t-1} + \mathbf{g}_{i_t} \cdot \tanh(\mathbf{W}_h \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h)$$

$$\mathbf{h}_t = \mathbf{g}_{o_t} \cdot \tanh(\mathbf{c}_t)$$

$$(\mathbf{h}_t, \mathbf{c}_t) := f_{ch} (\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t; \theta_{ch})$$

V)

LSTM : vue globale



Remarque : LSTM bidirectionnel = même chose que pour le RNN bidirectionnel

V)

« Gated Recurrent Unit » (GRU)

$$\mathbf{g}_{r_t} = \sigma(\mathbf{W}_{g_r} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_r}) \quad \text{«reset gate»}$$

$$\mathbf{g}_{z_t} = \sigma(\mathbf{W}_{g_z} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_z}) \quad \text{«update gate»}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \begin{bmatrix} \mathbf{g}_{r_t} \cdot \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h)$$

$$\mathbf{h}_t = (1 - \mathbf{g}_{z_t}) \cdot \tilde{\mathbf{h}}_t + \mathbf{g}_{z_t} \cdot \mathbf{h}_{t-1}$$

- Plus « léger », pas de cellule, et moins de paramètres que LSTM
- Généralise le RNN « standard »

Si tous les éléments de la « reset gate » valent 1

Si tous les éléments de la « update gate » valent 0

→ RNN « standard »

V)

« Gated Recurrent Unit » (GRU)

$$\mathbf{g}_{r_t} = \sigma(\mathbf{W}_{g_r} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_r}) \quad \text{«reset gate»}$$

$$\mathbf{g}_{z_t} = \sigma(\mathbf{W}_{g_z} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{g_z}) \quad \text{«update gate»}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \begin{bmatrix} \mathbf{g}_{r_t} \cdot \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_h)$$

$$\mathbf{h}_t = (1 - \mathbf{g}_{z_t}) \cdot \tilde{\mathbf{h}}_t + \mathbf{g}_{z_t} \cdot \mathbf{h}_{t-1}$$

$$\mathbf{h}_t := f_h (\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\theta}_h)$$

- Plus « léger », pas de cellule, et moins de paramètres que LSTM
- Généralise le RNN « standard »

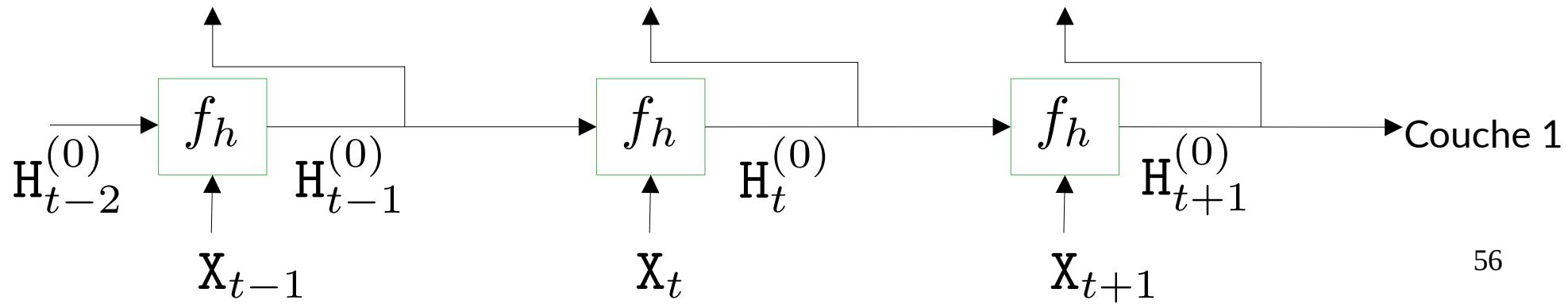
Si tous les éléments de la « reset gate » valent 1

Si tous les éléments de la « update gate » valent 0

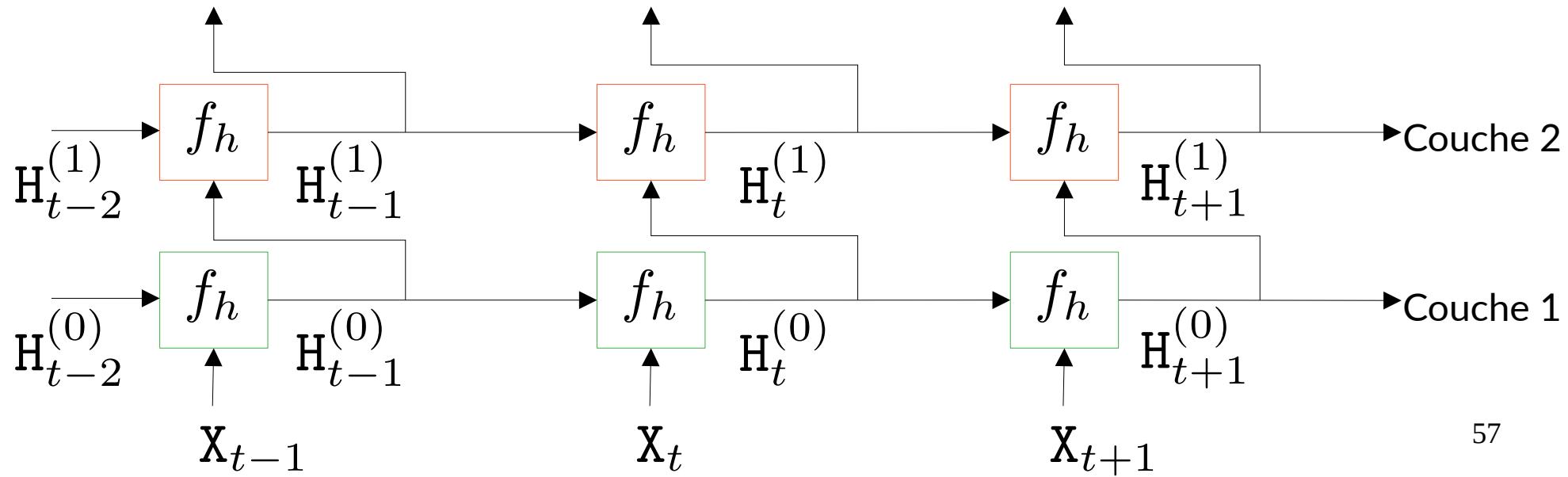
→ RNN « standard »

VI) Réseaux de neurones récurrents profonds (« Deep RNN »)

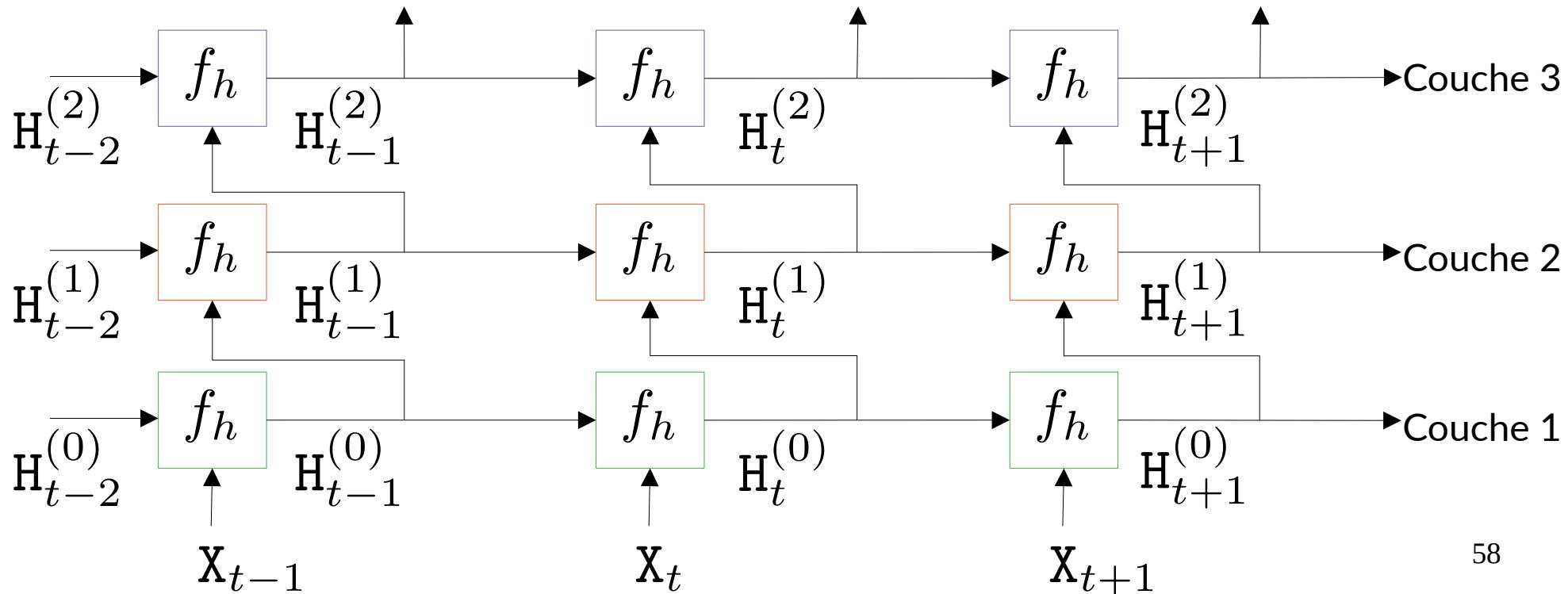
Réseaux de neurones récurrents profonds (« Deep RNN »)



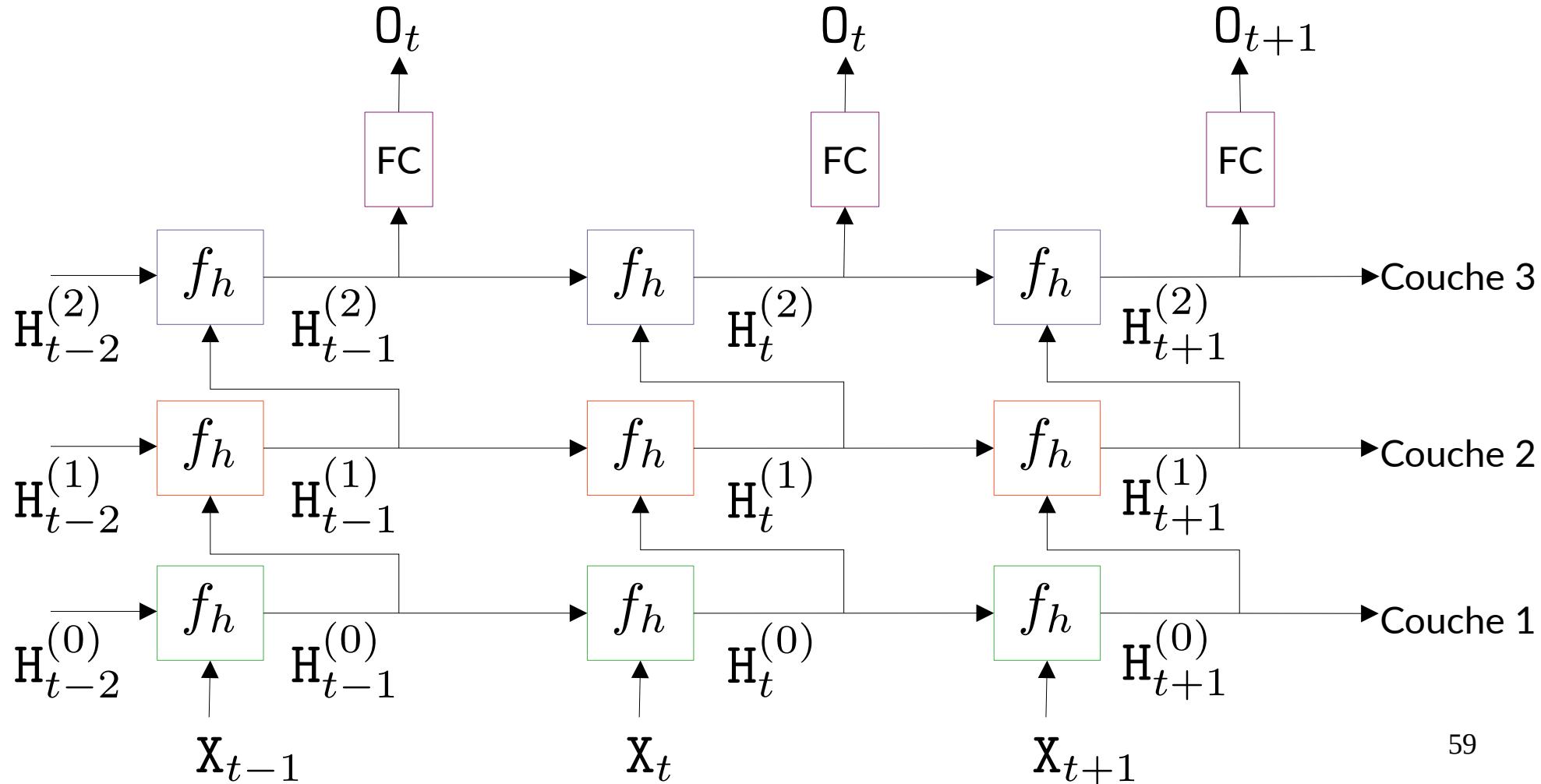
Réseaux de neurones récurrents profonds (« Deep RNN »)



Réseaux de neurones récurrents profonds (« Deep RNN »)



Réseaux de neurones récurrents profonds (« Deep RNN »)



VII) Limites et conclusion

Limites et conclusion

Avantages du RNN

- Peut s'appliquer à beaucoup de problèmes
- Peut théoriquement apprendre de très longues dépendances

Limites et conclusion

Avantages du RNN

- Peut s'appliquer à beaucoup de problèmes
- Peut théoriquement apprendre de très longues dépendances

Inconvénients du RNN

- Séquentiel par nature
 - Peu parallélisable
 - Apprentissage long

Limites et conclusion

Avantages du RNN

- Peut s'appliquer à beaucoup de problèmes
- Peut théoriquement apprendre de très longues dépendances

Inconvénients du RNN

- Séquentiel par nature
 - Peu parallélisable
 - Apprentissage long

RNN souvent remplacés par les CNN et dernièrement par les Transformers
Mais certaines idées semblent revenir : « Less is More: Recursive Reasoning with Tiny Networks »

Annexe : Token, Vocabulaire et Dictionnaire

1) Vocabulaire = Tokeniseur =

transformer un morceau de texte (« token ») en entier (« token ID »)

ex : un token = une lettre, un token = un mot, ou un token = des octets (« byte-pair encoding »)

	Token = « morceau de texte »	Token ID = Entier
lettre	« c »	2
mot	« chat »	2307
Octets (« BPE »)	« cha »	243

Annexe : Token, Vocabulaire et Dictionnaire

1) Vocabulaire = Tokeniseur =

transformer un morceau de texte (« token ») en entier (« token ID »)

ex : un token = une lettre, un token = un mot, ou un token = des octets (« byte-pair encoding »)

2) Dictionnaire =

transformer un entier (« token ID ») en un vecteur de dimension D (« embedding vector »)

	Token = « morceau de texte »	Token ID = Entier	Embedding vector = Vecteur de dimension D
lettre	« c »	2	[0.12, -0.41, 0.47, ...]
mot	« chat »	2307	[0.90, -0.53, -0.18, ...]
Octets (« BPE »)	« cha »	243	[-0.67, -0.09, 0.93, ...]

Annexe : Token, Vocabulaire et Dictionnaire (suite)

« one-hot encoding »

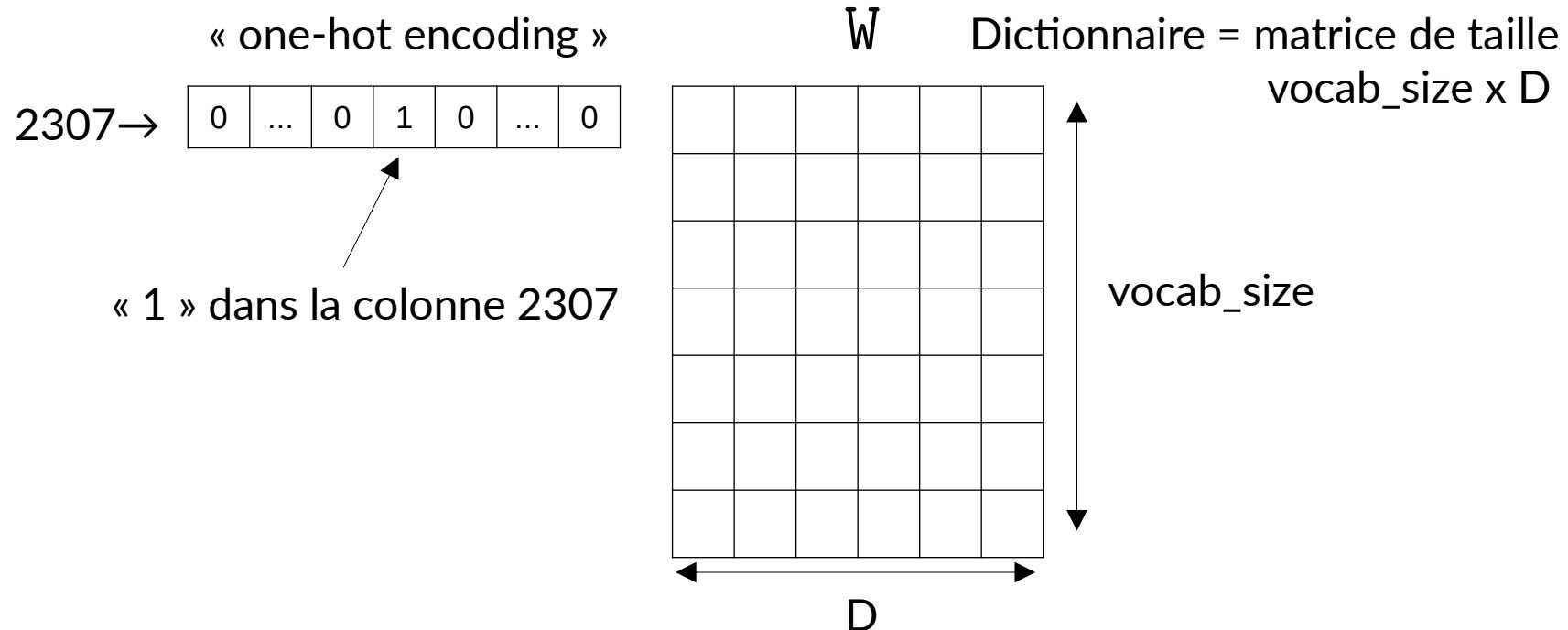
2307 →

0	...	0	1	0	...	0
---	-----	---	---	---	-----	---

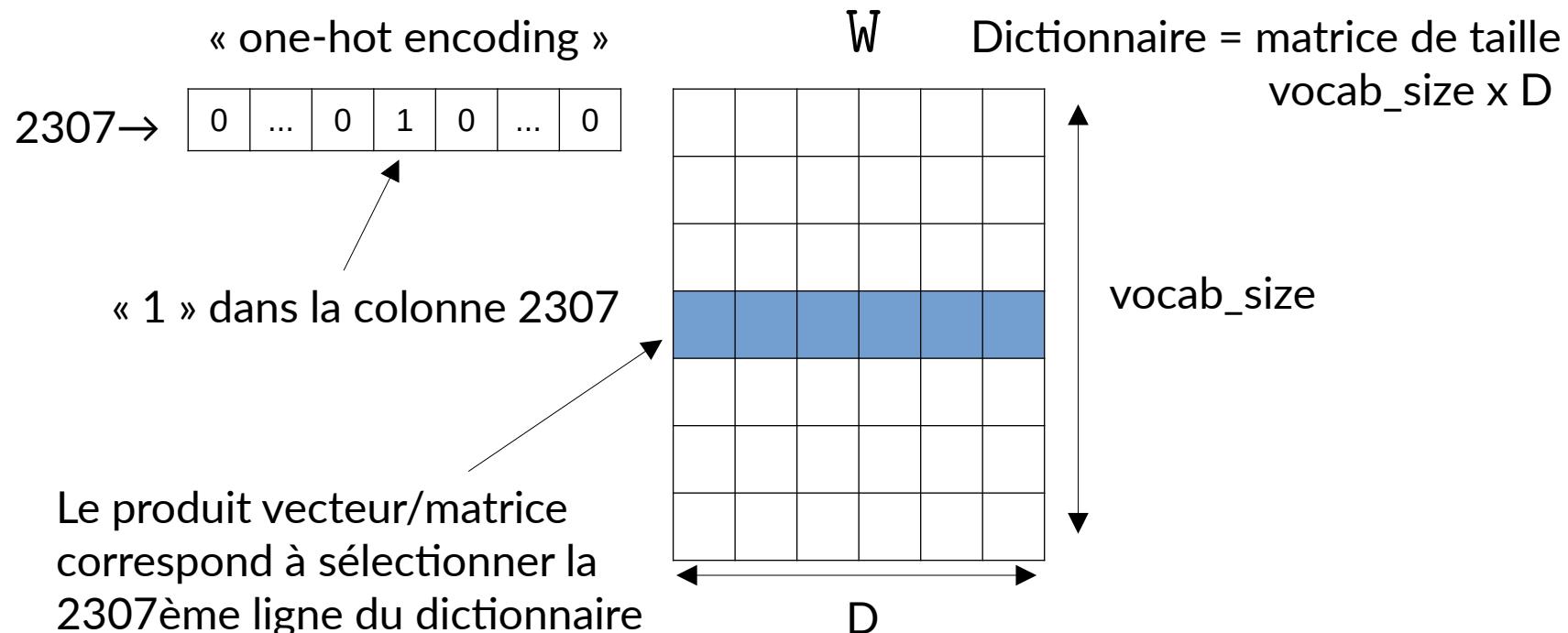


« 1 » dans la colonne 2307

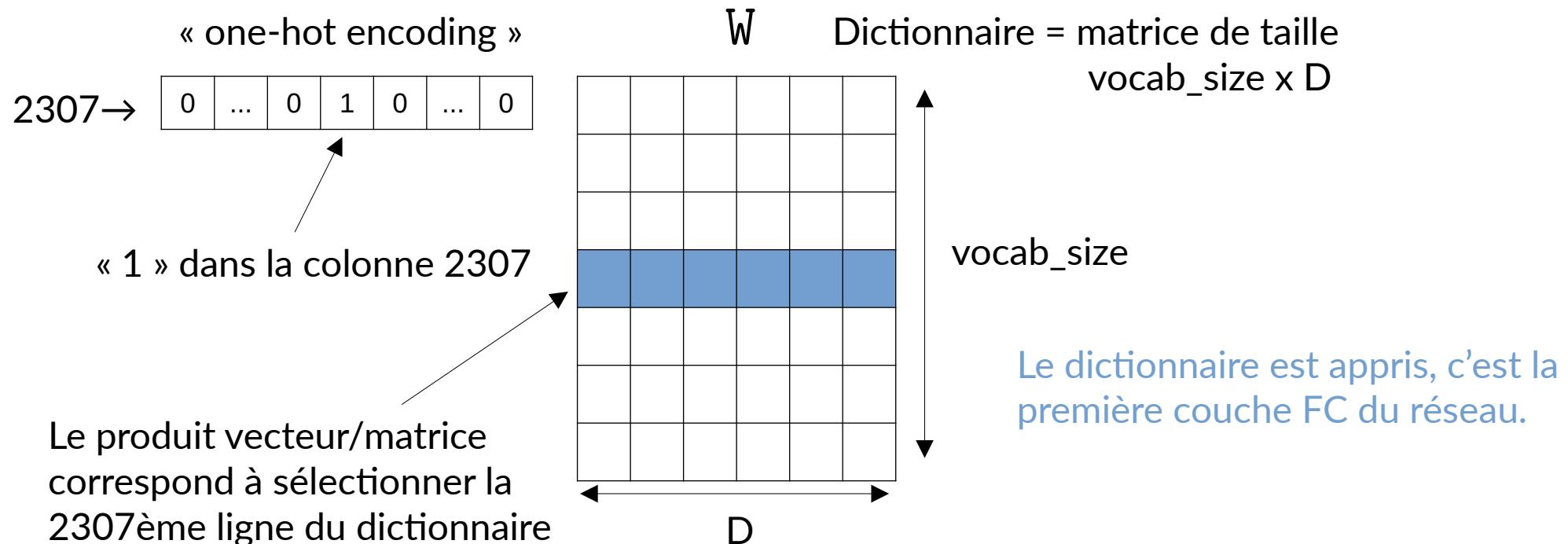
Annexe : Token, Vocabulaire et Dictionnaire (suite)



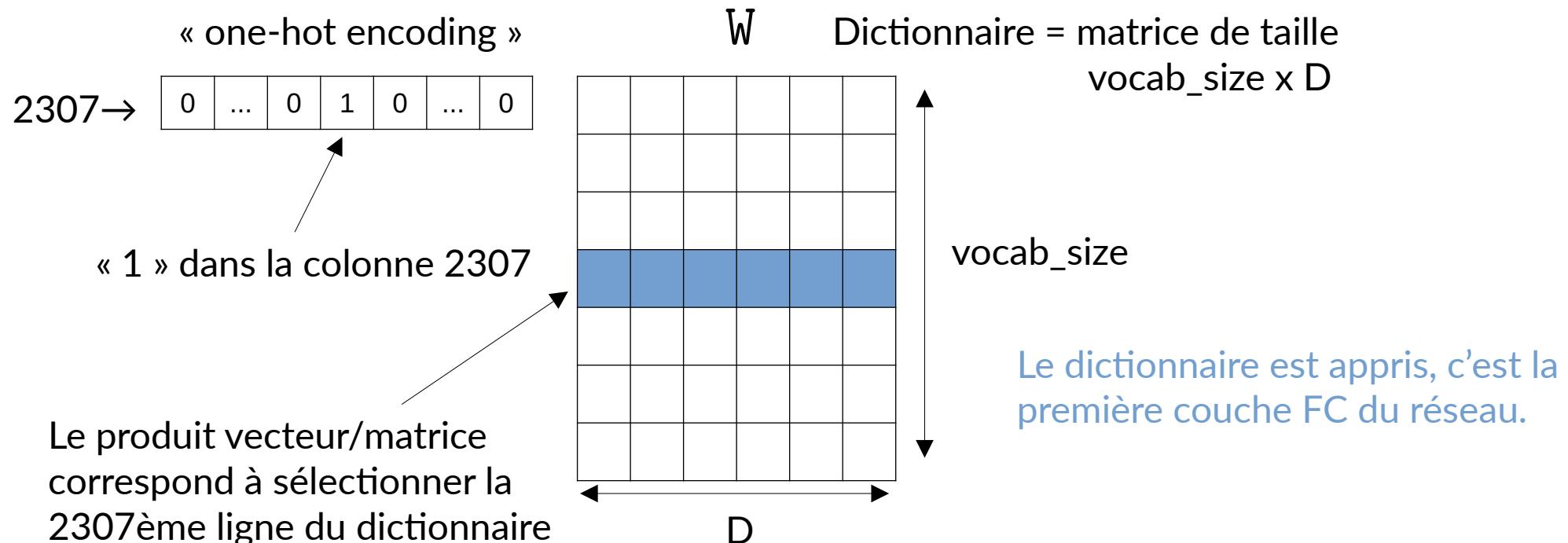
Annexe : Token, Vocabulaire et Dictionnaire (suite)



Annexe : Token, Vocabulaire et Dictionnaire (suite)



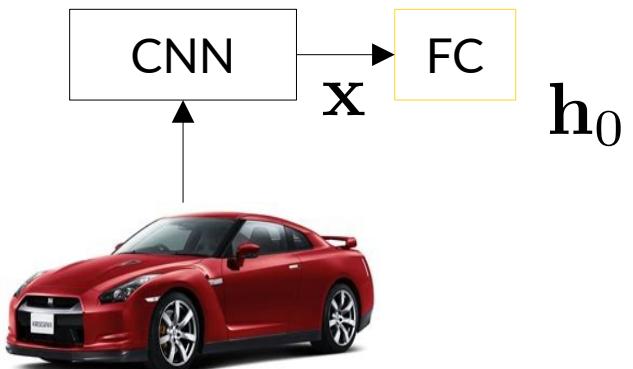
Annexe : Token, Vocabulaire et Dictionnaire (suite)



Implémentation :

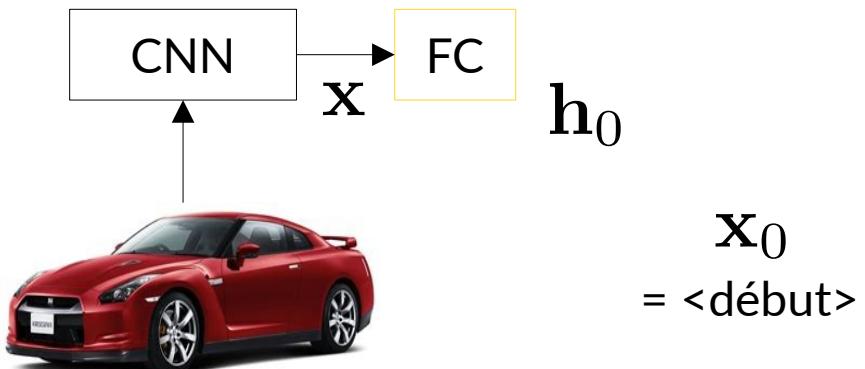
- A) La 1ère couche d'un réseau est une FC sans biais de taille $\text{vocab_size} \times D$
→ très inefficace ! Il faut créer le vecteur « one-hot » puis faire le produit matriciel.
- B) Remplacer la 1ère couche FC du réseau par une implémentation efficace qui prend en entrée le token ID et qui sélectionne la ligne correspondante du dictionnaire.

TP : Description d'une image (inférence)

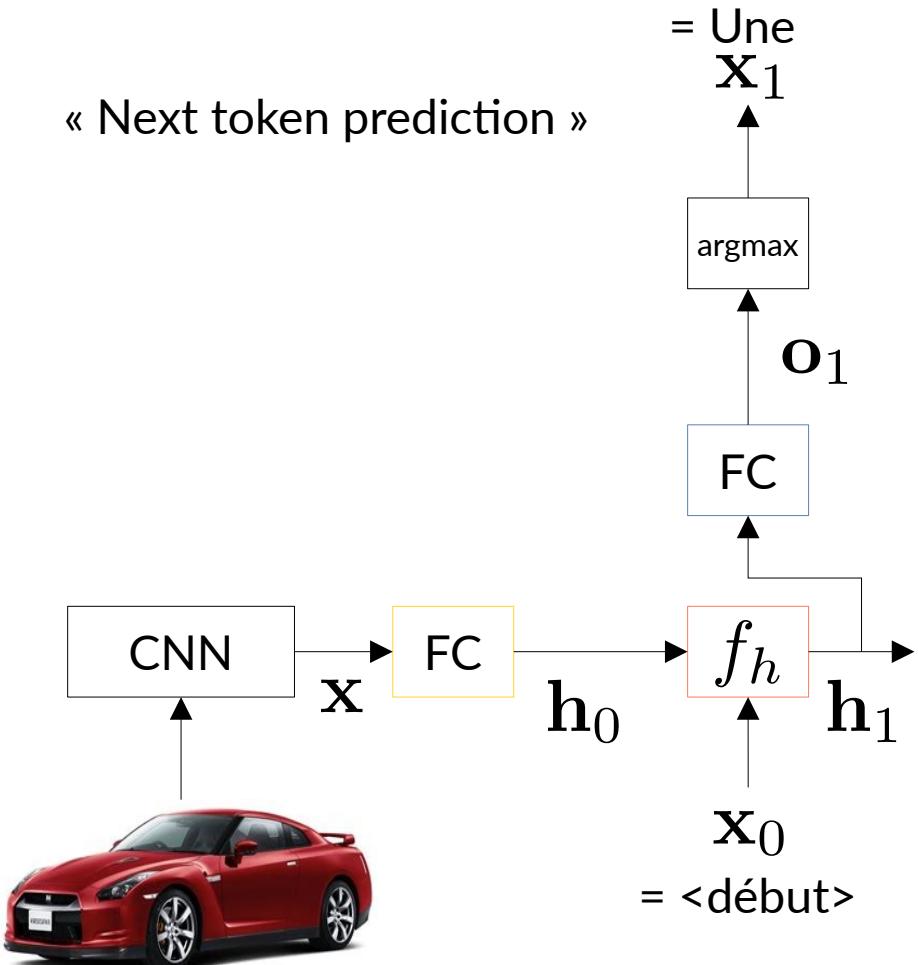


TP : Description d'une image (inférence)

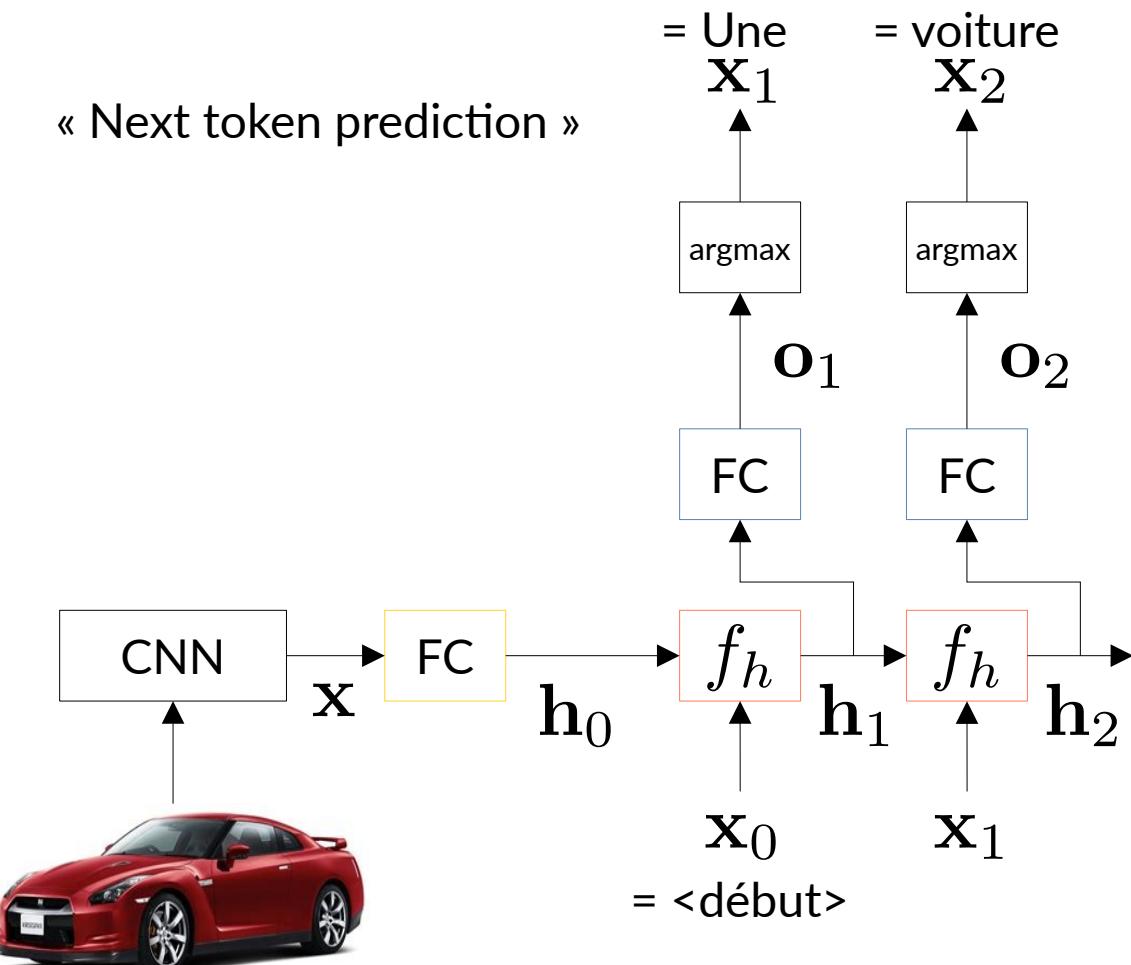
« Next token prediction »



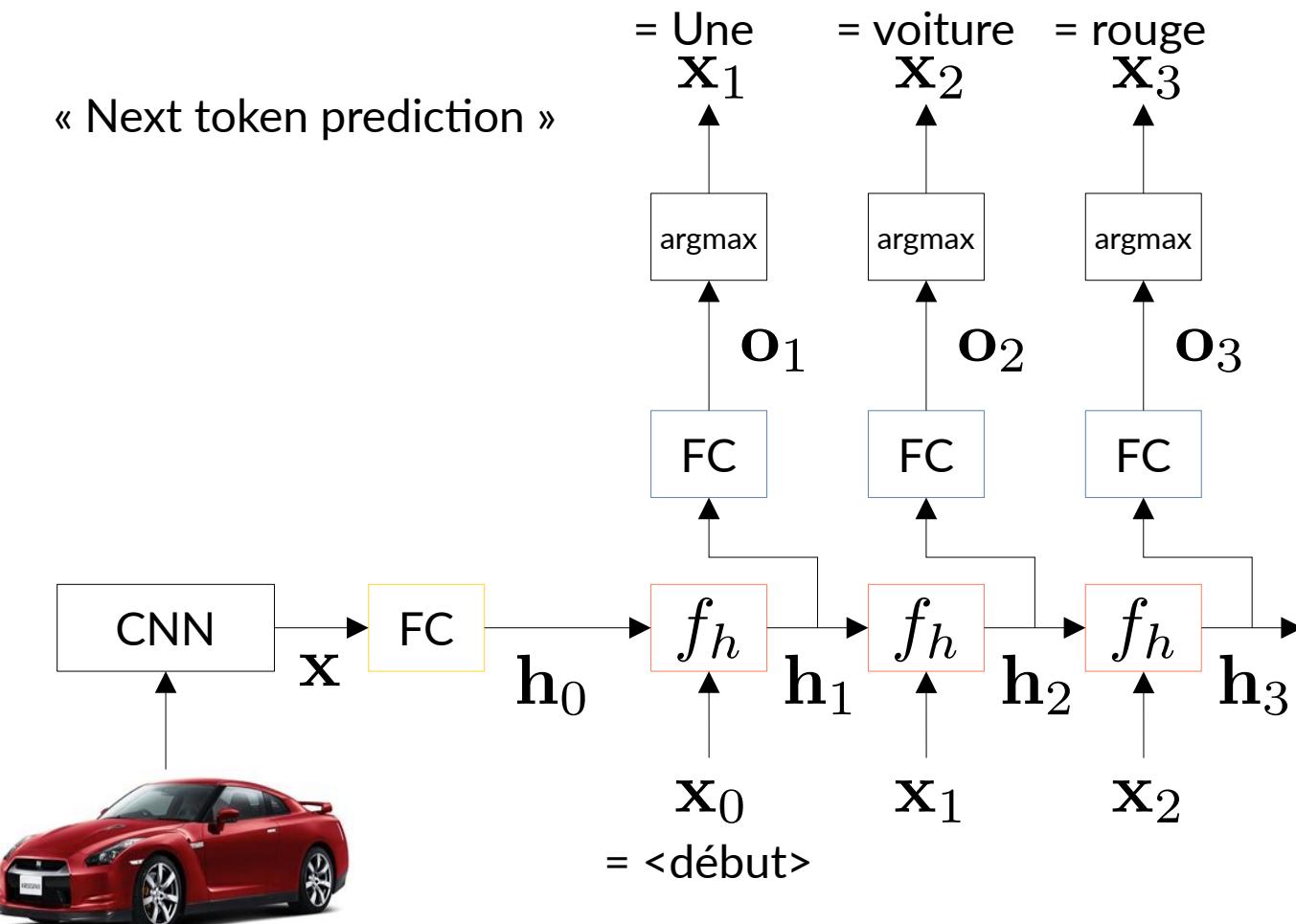
TP : Description d'une image (inférence)



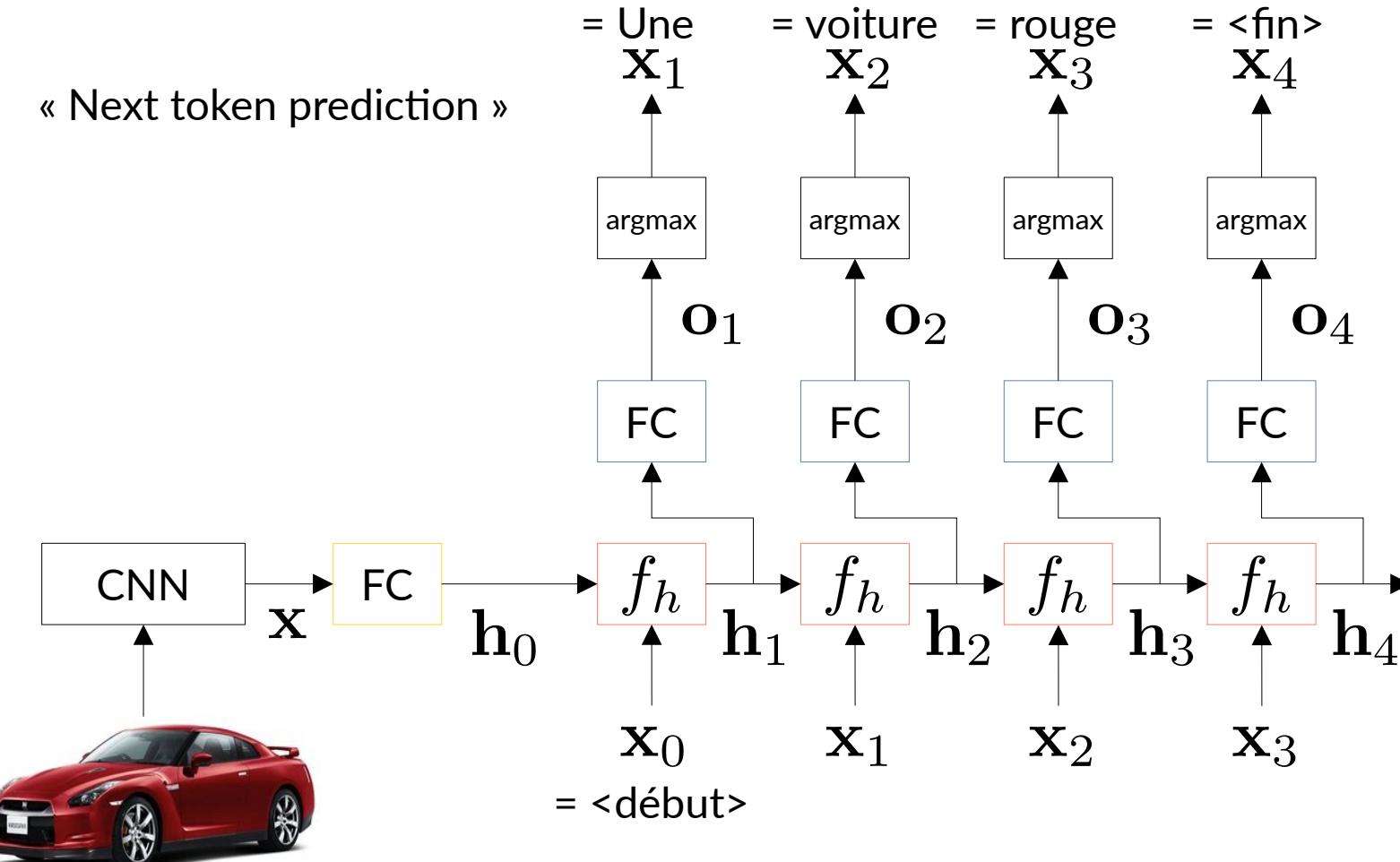
TP : Description d'une image (inférence)



TP : Description d'une image (inférence)

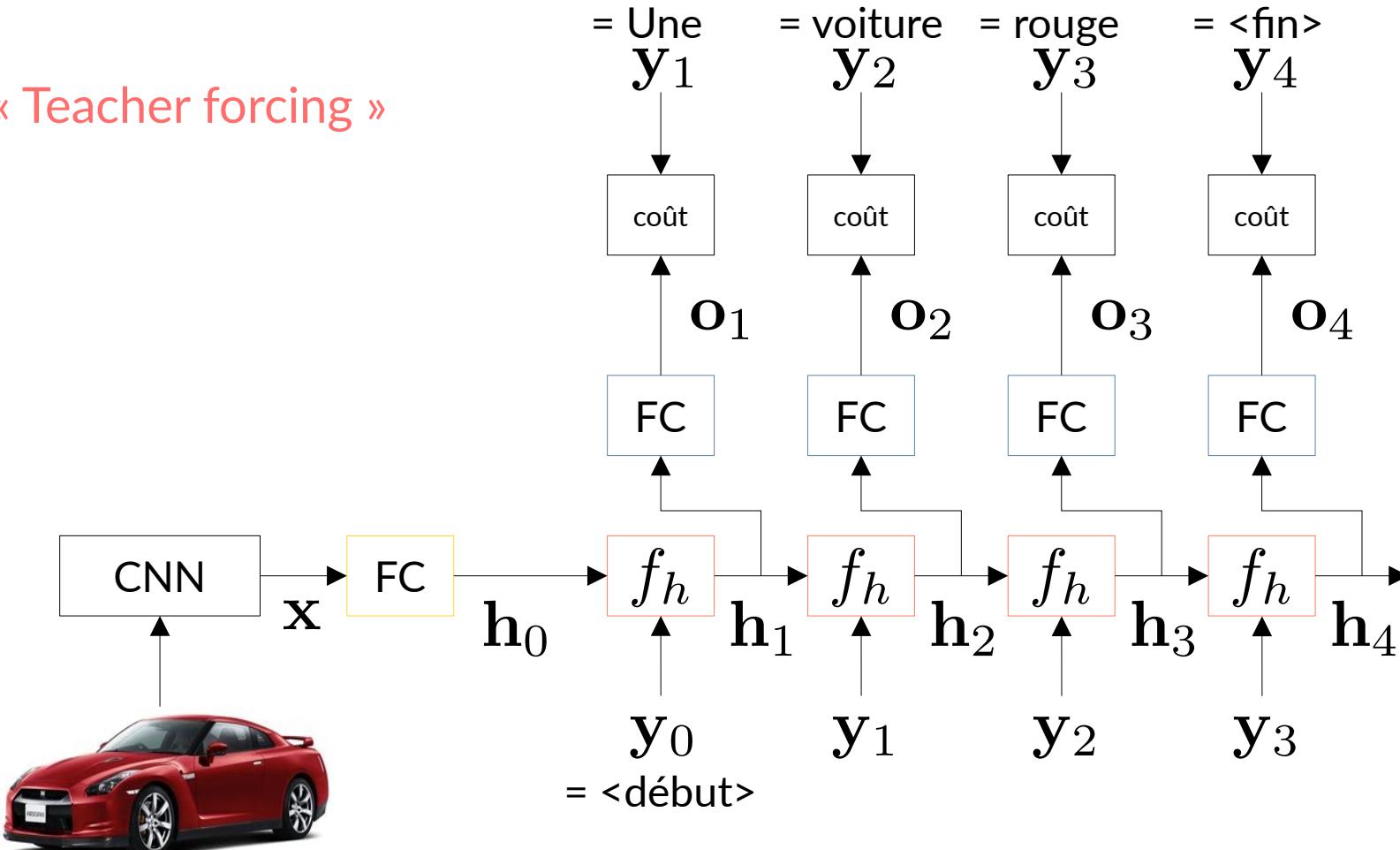


TP : Description d'une image (inférence)



TP : Description d'une image (apprentissage)

« Teacher forcing »



TP : Description d'une image (apprentissage)

« Teacher forcing »

L'entrée à l'instant t
n'est pas la prédiction à
l'instant $t-1$ mais
l'étiquette à l'instant $t-1$.



CNN

FC

h_0

y_0
= <début>

FC

FC

FC

FC

h_1

h_2

h_3

h_4

coût

coût

coût

coût

y_1
= Une

y_2
= voiture

y_3
= rouge

y_4
= <fin>