Détection des parenthèses dans un LSTM bilingue : Neurone partagé ou mécanisme distribué ?

Gwendal Tsang
Nanterre Université
gwendal.tsang@gmail.com

Anastasiia Belosevich
Université Sorbonne Nouvelle
a.belosevich@gmail.com

Résumé

Les LSTM développent spontanément des mécanismes de gestion de la syntaxe lors de l'entraînement sur du langage naturel. Cet article examine si un modèle LSTM entraîné sur un corpus bilingue (anglais et français) alloue des neurones distincts pour chaque langue ou s'il développe une représentation partagée pour des structures syntaxiques universelles comme les parenthèses. Nos expériences montrent l'émergence d'un unique neurone capable de détecter le statut *inside/outside* des parenthèses indépendamment de la langue, avec un score de détermination R^2 de 0.6971, soutenant l'hypothèse de la compression des représentations dans les modèles multilingues.

1 Introduction

Karpathy et al. (2015) ont montré, en visualisant les activations des cellules d'un LSTM, que certains neurones se spécialisent spontanément dans des tâches spécifiques, comme le suivi de la longueur des lignes ou la gestion des paires de guillemets et de parenthèses [1].

De nombreuses études ont reproduit ces résultats. Mais, à notre connaissance, aucune étude n'a testé ce phénomène d'émergence d'un module syntaxique spécialisé et localisé dans un LSTM entraîné simultanément sur deux langues avec un vocabulaire partagé.

La question centrale de cette étude est la suivante : dans un contexte bilingue (anglais/français) utilisant un vocabulaire partagé (au niveau des caractères), comment le modèle encode-t-il une règle syntaxique universelle ?

Nous formulons deux hypothèses mutuellement exclusives :

— **Hypothèse de séparation** : le réseau utilise le contexte linguistique h_{t-1} pour diriger le signal vers des neurones différents selon la langue (différents, ou à tout le moins fortement séparés).

— **Hypothèse de chevauchement** : la forte similarité de l'entrée x_t (le caractère `(` ayant le même embedding dans les deux langues) et l'identité fonctionnelle de la parenthèse forcent l'utilisation d'un mécanisme partagé. Ainsi le LSTM converge vers un mécanisme unique et partagé, invariant à la langue.

L'ensemble du code source ainsi que les corpus utilisés pour l'entraînement et le probing sont mis à disposition sur GitHub : github.com/GwenTsang/Neural_networks_interpretability

1. Karpathy et al. (2015) : « (...) one [LSTM's] cell is clearly acting as a line length counter, starting with a high value and then slowly decaying with each character until the next newline. Other cells turn on inside quotes, the parenthesis after if statements, inside strings or comments, or with increasing strength as the indentation of a block of code increases. »

2 Méthodologie

2.1 Architecture du modèle

Nous utilisons un LSTM au niveau des caractères (*CharLSTM*). Pour favoriser l'interprétabilité et éviter la dilution de l'information (phénomène de *polysemanticity*), nous avons opté pour une architecture restreinte :

— **Embedding :** 128 dimensions.
— **Couches cachées :** 1 couche unique de 512 neurones.
— **Vocabulaire :** 115 caractères (partagé entre les deux langues).

Le choix d'une seule couche est motivé par nos observations expérimentales : les configurations à deux couches tendent à disperser l'information, réduisant le score R^2 local maximal observable (ex. $R^2 \approx 0.475$ pour 2 couches contre ≈ 0.70 pour 1 couche).

2.2 Données et entraînement

Le corpus d'entraînement est composé d'un mélange de textes anglais et français (incluant l'intégrale de Flaubert et d'autres textes littéraires). Le modèle est entraîné sur la tâche de prédiction du caractère suivant (*Next Character Prediction*). Les hyperparamètres retenus après recherche sont : learning rate 8.8×10^{-4}, dropout ≈ 0.17, et une séquence de longueur 256.

2.3 Protocole de *probing*

Pour identifier les neurones spécialisés, nous extrayons les états de cellule c_t (*cell state*) de la dernière couche. Nous définissons une tâche de régression linéaire univariée (Ridge) pour prédire si le caractère courant est à l'intérieur d'une parenthèse.

Soit $y_t \in \{0, 1\}$ l'étiquette binaire indiquant la présence à l'intérieur de parenthèses au temps t, et $c_{t,j}$ l'activation du j-ème neurone de la cellule. Nous cherchons le neurone j^* qui maximise le coefficient de détermination R^2 sur un jeu de test bilingue :

$$j^* = \underset{j}{\operatorname{argmax}}\, R^2(y, c_{:,j}) \tag{1}$$

Cette méthode permet de détecter des neurones « mono-sémantiques » (dédiés) plutôt que des représentations distribuées.

Régression Ridge univariée. Pour chaque neurone $j \in \{1, \ldots, 512\}$, on résout indépendamment :

$$\min_{w_j} \sum_{i=1}^{N} (y_i^c - w_j\, x_{i,j}^c)^2 + \alpha\, w_j^2 \tag{2}$$

où x^c et y^c sont centrés, et $\alpha = 10$.

3 Résultats

L'analyse par *probing* sur un jeu de données de test (14 362 échantillons) a permis d'identifier le neurone d'index 250 comme le neurone dont l'activité est la plus fortement corrélée à la propriété booléenne « être entre parenthèses ». La visualisation du graphe d'activité de ce neurone suggère qu'il s'active à l'ouverture d'une parenthèse et redescend lorsque la parenthèse se ferme :

Soient c_t la mémoire du LSTM et f_t la *forget gate*. Dans la mise à jour standard du LSTM, on a notamment le terme

$$f_t \odot c_{t-1}.$$

On constate facilement que si f_t reste proche de 1, la mémoire est conservée. À l'inverse, plus f_t est proche de 0, plus la mémoire du LSTM est effacée.

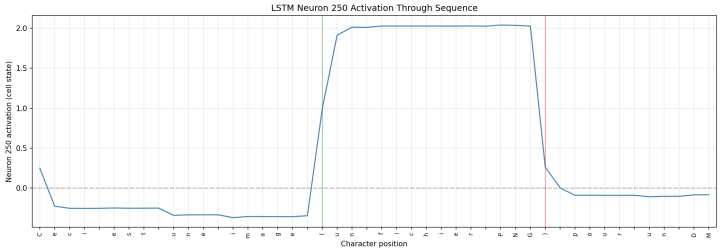


FIGURE 1 – Activations dans le temps du neurone 250 sur une phrase

Pour un neurone donné (ici le 250), la porte d'entrée est :

$$i_t^{(250)} = \sigma\left(\underbrace{W_{ii}^{(250)} \cdot x_t}_{\text{effet du token}} + \underbrace{W_{hi}^{(250)} \cdot h_{t-1}}_{\text{effet du contexte}} + b_i^{(250)}\right).$$

Où $W_{ii}^{(250)}$ et x_t (l'embedding du token) sont deux vecteurs. Leur produit scalaire est l'effet direct du token sur l'ouverture de la porte.

3.1 Ablation du neurone 250

Afin de déterminer si le neurone 250 joue un rôle causal dans la représentation du statut "à l'intérieur des parenthèses" nous menons une série d'expériences d'ablation consistant à neutraliser sélectivement ce neurone (sa composante $c_{t,250}$ plus précisement) et à mesurer l'impact sur les performances de décodage.

Nous effectuons des interventions au moment de l'inférence sur l'état de cellule c_t :

— **Lésion** : forcer $c_{t,250} = 0$ à chaque pas de temps.

— **Clamp** : fixer $c_{t,250}$ à une constante K (valeur typique "à l'intérieur") ou à 0 ("à l'extérieur").

Nous mesurons ensuite la quantité d'information restante concernant le label binaire *inside/outside* à l'aide d'une sonde linéaire (Ridge) appliquée soit à l'ensemble des 512 neurones (représentation distribuée, R^2_{dist}), soit au seul neurone 250 (représentation locale, R^2_{local}).

```
================================================
SUMMARY
================================================
Condition            R² local     R² distributed
------------------------------------------------
Baseline             0.4744       0.6674
Lesion (=0)          -0.0001      0.5855
Clamp=0              -0.0001      0.5855
Clamp=K              -0.0001      0.5788
================================================
```

TABLE 1 – Performances des sondes linéaires sous différentes conditions d'ablation

La lésion du neurone 250 entraîne une chute de $\Delta R^2_{\text{dist}} \approx 0.082$, tandis que la lésion de neurones aléatoires (Figure 2) n'induit qu'une variation négligeable ($\Delta R^2_{\text{dist}} \approx 3 \times 10^{-4}$ en moyenne, $n = 20$). Cette différence identifie le neurone 250 comme un contributeur causal à la représentation apprise, et non comme un simple marqueur passif.

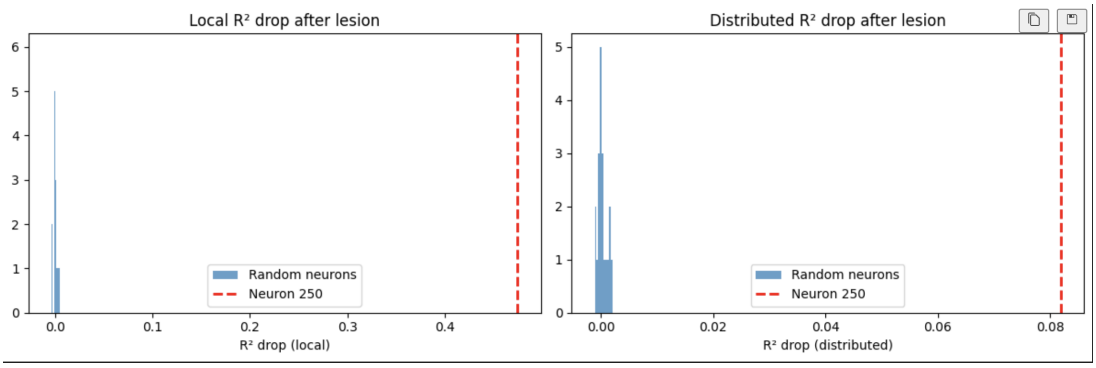


FIGURE 2 – Distribution des chutes de R^2 après lésion de neurones aléatoires comparée à la lésion du neurone 250.

Par ailleurs, le clampage (Figure 3) à une valeur constante (qu'elle soit 0 ou K) élimine toute prédictibilité locale ($R^2_{\text{local}} \approx 0$), démontrant que l'interprétabilité du neurone provient de sa capacité à basculer entre états, plutôt que d'une corrélation statique.

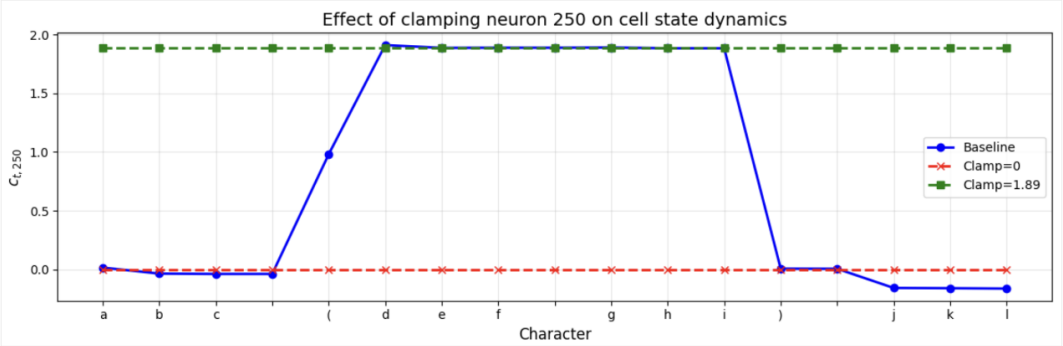
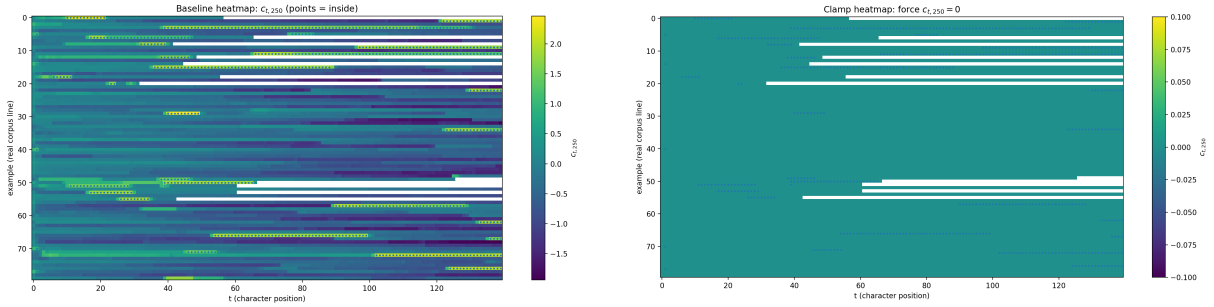


FIGURE 3 – Effet du clampage du neurone 250 sur la dynamique de l'état de cellule

Afin de compléter cette analyse quantitative, nous visualisons la dynamique temporelle de la composante $c_{t,250}$ sur des phrases réelles issues du corpus d'évaluation, à l'aide de heatmaps (Figure 4). En condition baseline, $c_{t,250}$ adopte un régime stable sur les intervalles compris entre les parenthèses ouvrantes et fermantes, tandis que des transitions nettes apparaissent aux frontières structurelles. Cette organisation temporelle correspond précisément aux régions annotées comme *inside*, indiquant que le neurone encode un état de mémoire maintenu dans le temps plutôt qu'une simple réponse ponctuelle aux symboles.

À l'inverse, lorsque $c_{t,250}$ est clampée à zéro à chaque pas de temps, cette structure disparaît entièrement : la heatmap devient homogène et ne présente plus aucune corrélation avec le statut *inside*/*outside*. Cette perte complète de structure temporelle confirme que l'information portée par le neurone 250 repose sur sa capacité à basculer dynamiquement entre états, et non sur un niveau d'activation statique. Les visualisations renforcent ainsi l'interprétation causale issue des expériences d'ablation.



(a) Baseline **(b)** Clamp $c_{t,250} = 0$

FIGURE 4 – Heatmaps de la composante $c_{t,250}$ sur des phrases réelles.

4 Analyse théorique

Le comportement du neurone 250 s'explique par la dynamique interne des portes du LSTM, dont le schéma détaillé et les équations sont présentés en **Annexe 1**. À la lumière des poids analysés et du diagramme fonctionnel, nous interprétons le mécanisme comme suit :

1. **Phase d'injection `'('` :** Comme vu dans les résultats, le caractère `(` possède une forte projection positive sur les poids d'entrée. Cela sature la *porte d'entrée* ($i_t \approx 1$), permettant au signal candidat $\tilde{c}_t$ d'être ajouté à l'état de la cellule. La porte d'oubli f_t demeure proche de 1, garantissant que l'information ainsi encodée sera conservée sur les pas de temps suivants.
2. **Phase de mémorisation :** Lorsque le modèle parcourt le contenu strictement entre parenthèses, les caractères n'activent pas la porte d'entrée ($i_t \approx 0$). Il est d'intérêt crucial que la *porte d'oubli* f_t reste saturée à 1 pour que le neurone agisse / déclenche comme mémoire persistante, insensible au contenu lexical et principalement sensible à la structure. Ainsi, cela permet à l'information c_{t-1} de persister presque intacte ($c_t \approx 1 \cdot c_{t-1} + 0$).
3. **Phase de réinitialisation `')'` :** La fermeture déclenche une réinitialisation de l'état interne, soit par une diminution brutale de f_t, soit par l'injection d'un signal de signe opposé via i_t, conduisant à une annulation rapide de c_t. Le neurone repasse ainsi dans un régime interepretable par "ne pas être entre des parenthèses".

Cette interprétation est confirmée par nos expériences d'intervention : le clampage de $c_{t,250}$ à une valeur constante supprime toute structure temporelle, et la lésion du neurone dégrade significativement la prédiction du statut inside/outside. L'information repose donc sur la dynamique de bascule des portes et non sur un niveau d'activation statique.

Ce mécanisme confirme que le LSTM n'apprend pas simplement une corrélation statistique locale, mais exploite la bistabilité de ses portes pour implémenter un compteur binaire robuste, invariant à la langue du contenu imbriqué. Ce mécanisme de saturation rend la moyenne des activations « Intérieur » très distincte de celle « Extérieur », expliquant le haut score R^2.

5 Discussion

L'émergence de ce neurone 250 n'est pas un phénomène constant dans nos expérimentations et semble être favorisée sous certaines conditions spécifiques. Il est possible, par exemple, que si on augmentait considérablement la taille du LSTM (en nombre de neurones), cela diminuerait la probabilité de voir émerger ce neurone spécialisé, et favoriserait, a contrario, un encodage distribué.

5.1 Généralisation aux autres délimiteurs

Nous avons découvert que ce neurone 250 réagit non seulement aux parenthèses mais également aux crochets selon la même logique d'activation-désactivation :

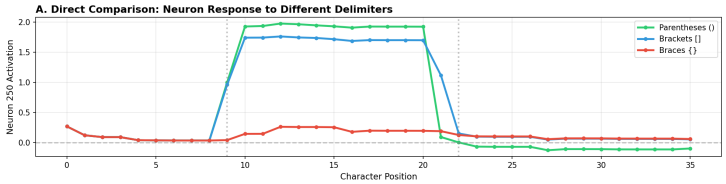


FIGURE 5 – Activations du neurone 250 sur trois types de délimiteurs

Nous avons analysé les poids internes du LSTM afin d'identifier le mécanisme causal pilotant l'activation du neurone 250. Nous avons calculé la projection (produit scalaire) des embeddings de divers délimiteurs sur les poids de la **porte d'entrée** (*input gate*) W_{ii} du neurone 250.

La porte d'entrée gouverne le flux d'information nouvelle entrant dans l'état de cellule. Une projection positive élevée indique qu'un token ouvre fortement la porte, déclenchant une mise à jour de la mémoire du neurone.

Résultats et corrélations Les projections révèlent une hiérarchie cohérente avec les intensités d'activation observées :

- **Déclencheur principal :** la parenthèse ouvrante `(` obtient une projection maximale de $+\mathbf{8.875}$, confirmant qu'il s'agit de la caractéristique dominante pour ce neurone.
- **Déclencheur secondaire :** le crochet ouvrant `[` obtient une projection forte, mais plus faible, de $+\mathbf{5.256}$.
- **Inhibition :** les délimiteurs fermants et certains terminateurs (p. ex. `)`, `.`) obtiennent des projections négatives (p. ex. -1.96, -4.96), inhibant la porte d'entrée et réduisant les activations parasites.

5.2 Imbrications des délimiteurs

Nos expériences précédentes utilisent des chaînes contenant des parenthèses imbriquées, mais notre label de probing est binaire, indépendamment du niveau d'imbrication exact.

Nous avons donc testé si le neurone 250 encode la profondeur d'imbrication ou uniquement un état binaire. Sur des chaînes synthétiques de type `(((...)))`, la corrélation entre $c_{t,250}$ et la profondeur est faible ($r = 0.19$, $p = 0.30$). L'activation reste stable (≈ 0.5) aux niveaux intermédiaires et n'augmente (≈ 1.3) qu'au niveau le plus profond. Le neurone 250 fonctionne donc comme un **détecteur binaire** (inside/outside), et non comme un compteur de profondeur.

Chaîne	depth=1	depth=2	depth=3	depth=4
`(def)`	1.334	—	—	—
`((def))`	0.490	1.381	—	—
`(((def)))`	0.491	0.501	1.350	—
`((((def))))`	0.491	0.501	0.500	1.310

TABLE 2 – Activation moyenne $c_{t,250}$ en fonction de la profondeur des parenthèses

Un corpus plus large ne modifierait probablement pas cette conclusion : les activations observées sont relativement stables, et la corrélation reste basse non par manque de données, mais parce que la relation est structurellement non linéaire. Même significative, une corrélation de $r = 0.19$ n'expliquerait que $\sim$4% de la variance ($r^2 = 0.036$).

Des travaux futurs pourraient néanmoins explorer si d'autres neurones encodent la profondeur (via un probing avec y_t = depth sur l'ensemble des 512 neurones), ou tester des imbrications plus profondes (5-6 niveaux) afin de vérifier si le mécanisme sature.

5.3 Utiliser un corpus d'entraînement plus structuré ?

Dans KARPATHY et al. (2015), les auteurs mènent également l'expérimentation sur un LSTM entraîné sur du code C (noyau Linux). Le code C contient des tokens structurels fréquents, réguliers et fortement corrélés (accrochages d'accolades, paires de parenthèses, indentations, mots-clés). Un corpus littéraire comparable contient moins de répétitions systématiques de ce type (`{}/()`), donc le signal supervisant pour une cellule « parenthèses » y est en général plus faible. C'est la raison pour laquelle nous avons opéré en amont un léger filtrage, dans l'optique d'augmenter la fréquence des parenthèses dans le corpus d'entraînement du LSTM. Il serait intéressant de mesurer la corrélation entre la fréquence des parenthèses et la probabilité d'obtenir un tel neurone spécialisé. Pour ce faire, il est possible d'ajouter des parenthèses synthétiquement et de relancer l'entraînement.

6 Conclusion

Cette étude confirme qu'il est possible de déterminer des conditions d'entraînement dans lesquelles un CharLSTM bilingue alloue spontanément un neurone unique pour gérer l'imbrication des parenthèses, indépendamment de la langue du texte. Ce phénomène de partage neuronal soutient l'idée que les structures syntaxiques invariantes agissent comme des points d'ancrage translinguistiques dans l'espace latent des modèles de langage.

Annexe 1 — Diagramme d'une cellule LSTM

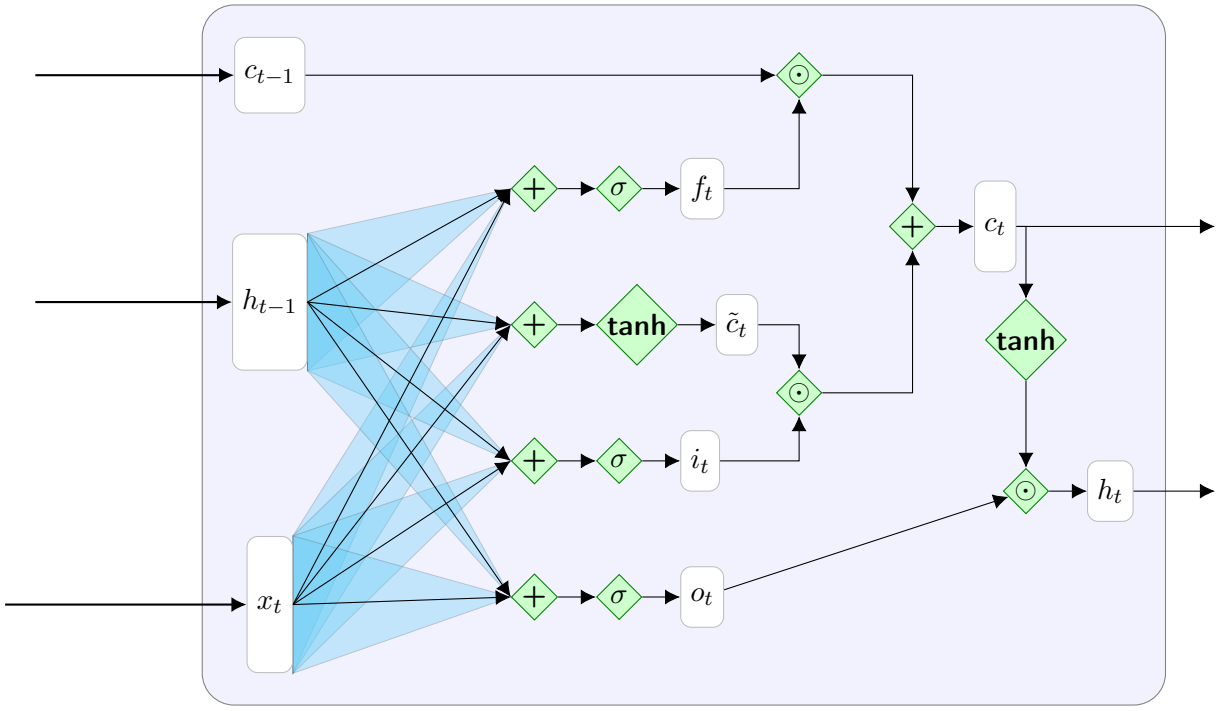


FIGURE 6 – Schéma d'une cellule LSTM au temps t inspiré de JURAFSKY et al. (2026)

- $\odot$ note le produit de Hadamard,
- h_{t-1} note l'état caché précédent,
- c_{t-1} note l'information des états précédents
- x_t correspond à au vecteur dense donné en entrée (vecteur associée à un token, qui est lui même associé à un caractère).

Les zones en cyan qui donnent sur des nœuds « + » symbolisent les transformations linéaires (matrices de poids) appliquées aux entrées. Quatre portes sont calculées :

— $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$: **porte d'oubli** (*forget gate*), contrôle la proportion de l'état de cellule précédent à conserver ;

— $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$: **porte d'entrée** (*input gate*), contrôle la proportion du candidat à intégrer ;

— $\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$: **candidat** (*cell candidate*), nouvelle information potentielle ;

— $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$: **porte de sortie** (*output gate*), contrôle la proportion de l'état de cellule à exposer.

L'état de cellule est ensuite mis à jour par $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$, puis l'état caché par $h_t = o_t \odot \tanh(c_t)$. Les flèches sortant du cadre indiquent que c_t et h_t sont transmis au pas suivant (avec x_{t+1}).

Références

JURAFSKY, Daniel et James H. MARTIN (2026). *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. Online manuscript released January 6, 2026. URL : https://web.stanford.edu/~jurafsky/slp3/.

KARPATHY, Andrej, Justin JOHNSON et Li FEI-FEI (2015). *Visualizing and Understanding Recurrent Networks*. arXiv : 1506.02078 [cs.LG]. URL : https://arxiv.org/abs/1506.02078.