Réseau de neurones récurrents

Introduction au deep learning

Théo Lopes Quintas

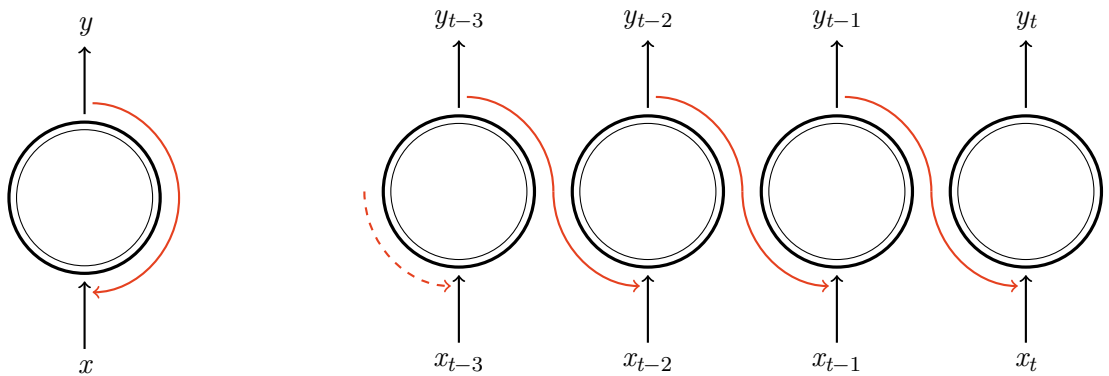
BPCE Payment Services,
Université Paris Dauphine

2023-2026

Architectures récurrente

Avec un neurone

Ici on a $y \in \mathcal{Y}$ et $x \in \mathbb{R}^d$, où d représente la somme des informations présentes et sa propre sortie du temps précédent : cela joue comme une mémoire.



(a) Schéma **(b)** Version dépliée

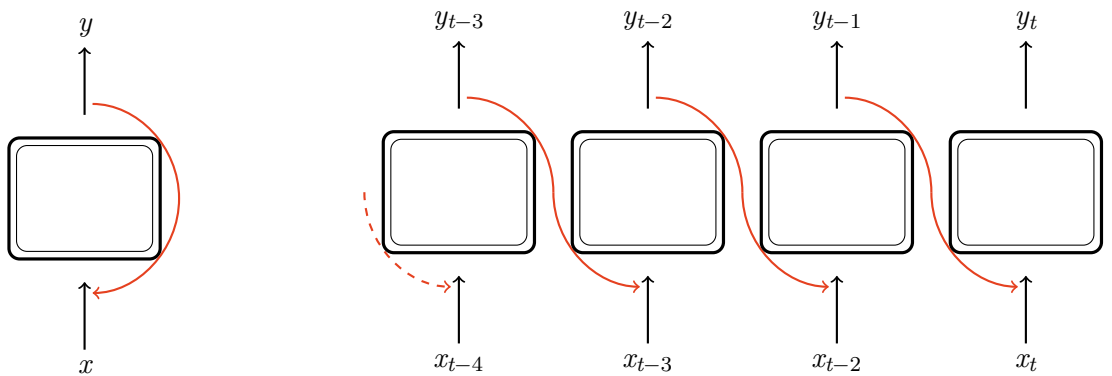
Figure – Réseau de neurone récurrent

Chaque neurone associe des poids aux d informations reçues, à la manière d'un neurone classique.

Architectures récurrente

Avec plusieurs neurones

Au lieu de considérer un unique neurones, on en considère n. On a maintenant $y \in \mathbb{R}^n$ et $x \in \mathbb{R}^d$, où d représente la somme des informations présentes et les n valeurs présentes dans la couche de neurones du temps précédent.



(a) Schéma **(b)** Version dépliée

Figure – Réseau de neurones récurrent

Dans la suite, on représentera le lien entre la sortie du temps $t-1$ à l'entrée du temps t par une flèche entre les deux rectangles.

Architectures récurrente

Série vers vecteur

On donne en entrée une série d'information et on obtient une unique valeur prédite.

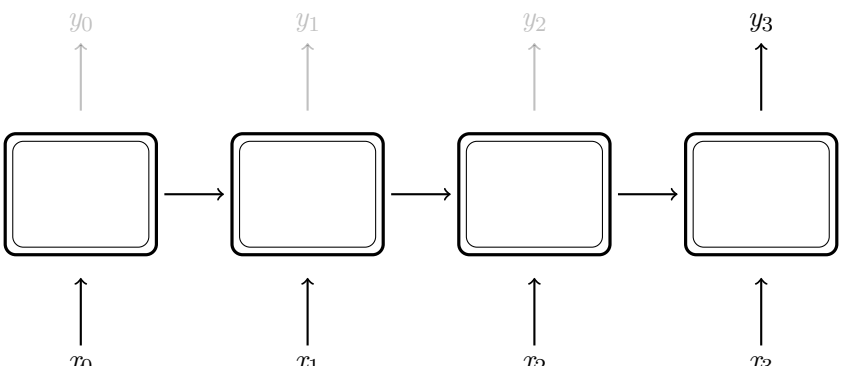


Figure – Réseau de neurones récurrents - Série vers vecteur

Dans le cadre d'une analyse de sentiment, on peut donner un texte critique et renvoyer une valeur entre -1 et 1 qui dit si on a détesté ou aimé.

Architectures récurrente

Vecteur vers série

En entrée on ne donne qu'une seule information et on obtient une série à partir de celle-ci.

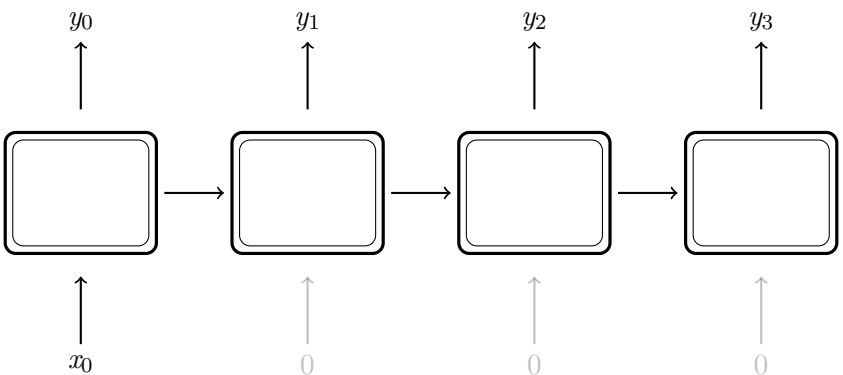


Figure – Réseau de neurones récurrents - Vecteur vers série

Si on donne une image en entrée, on peut obtenir une description de l'image par exemple.

Architectures récurrente

Série vers série

On donne cette fois une série d'informations et dans le même temps on reçoit une série.

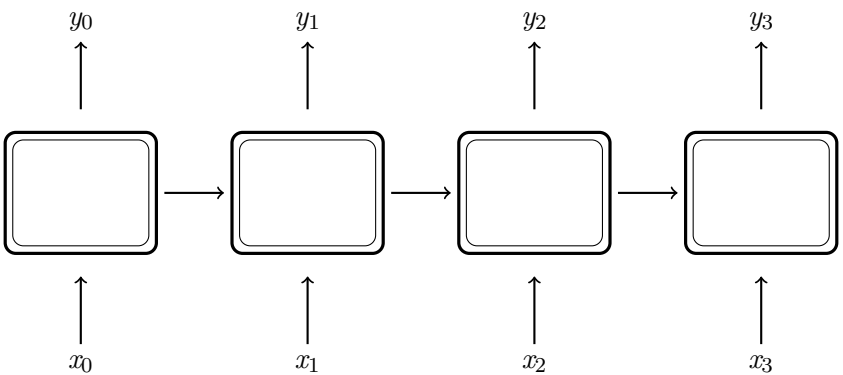


Figure – Réseau de neurones récurrents - Série vers série

Dans le cadre de prédiction financière, on peut renseigner l'ensemble des valeurs d'une action et demander la prédiction des valeurs décalées d'un jour : de la deuxième valeur d'entrainement jusqu'à la prochaine valeur inconnue pour le moment.

Architectures récurrente

Série vers série différée

On commence par considérer en entrée uniquement les inputs dans une partie *encoder* puis uniquement les sorties dans une partie *decoder*.

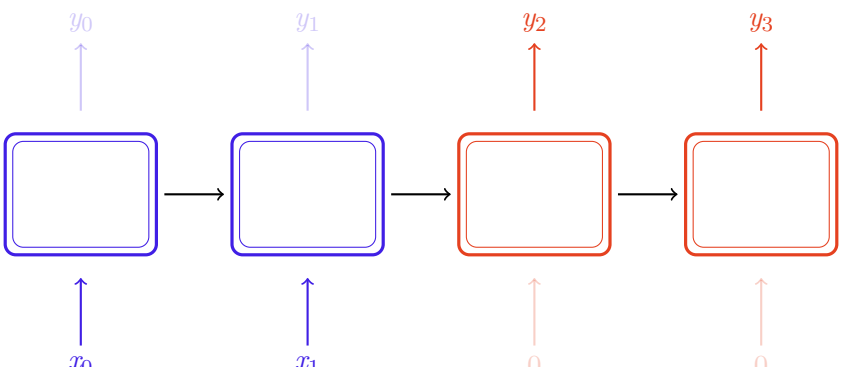


Figure – Réseau de neurones récurrents - Vecteur vers série

Dans le cadre d'une traduction, la fin de la phrase peut influencer la manière dont on va traduire le début. Il est donc plus efficace d'avoir attendu la fin de la série pour prédire la série.

En résumé

Cellules particulières

Cellule LSTM : Long Short Term Memory

La cellule LSTM a été proposé dans [Hochreiter and Schmidhuber, 1997] puis améliorée par [Sak et al., 2014] et [Zaremba et al., 2014] entre autre.

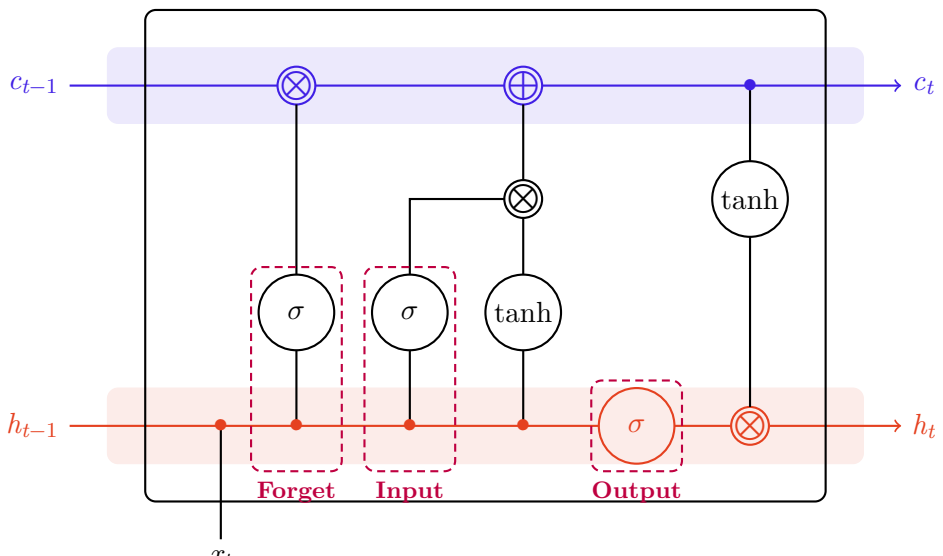


Figure – Schéma d'un LSTM avec état de cellule $(c_t)_{t\in\mathbb{N}}$ et état caché $(h_t)_{t\in\mathbb{N}}$

Cellules particulières

Cellule LSTM : Long Short Term Memory

La cellule LSTM peut être considérée comme une boîte noire et être utilisée comme une cellule récurrente au même titre qu'un neurone récurrent. Elle se compose de deux particularités.

Les deux états

Contrairement à un neurone récurrent classique, il y a deux vecteurs d'états stockés de manière indépendantes :

- **État caché** $(h_t)_{t\in\mathbb{N}}$: il correspond à l'état classique que se transmette les neurones récurrents
- **État de cellule** $(c_t)_{t\in\mathbb{N}}$: il correspond à une mémoire plus long terme de la cellule.

Ces deux vecteurs d'états sont mis à jour à travers un parcours plus complexe qu'un neurone récurrent classique.

Les trois portes

- **Forget** : prend l'état caché précédent et les informations puis passe par une fonction sigmoid pour modifier l'état de cellule en multipliant terme à terme
- **Input** : prends les mêmes informations que la porte *forget* et multiplie ces informations passées par une fonction sigmoid et tanh avant de l'ajouter à l'état de cellule
- **Output** : prends les mêmes informations que la porte *forget* et les passes dans la fonction sigmoid. En les multipliant par l'état de cellule passé dans la fonction tanh, on met à jour l'état caché

Cellules particulières

Cellule GRU : Gated Recurrent Unit

La cellule GRU [Cho et al., 2014] est une simplification de la cellule LSTM. Nous n'avons plus la partie de mémoire longue avec l'état de cellule.

GRU repose sur deux portes :

- **Reset** : même idée que la porte *forget* et *input*
- **Update** : similaire à la porte *output* en enlevant la partie tanh

En pratique il existe de nombreuses variations de la cellule LSTM. [Greff et al., 2016] propose une étude complète qui montre pour plusieurs architectures LSTM et plusieurs problèmes que toutes les variantes sont équivalentes.

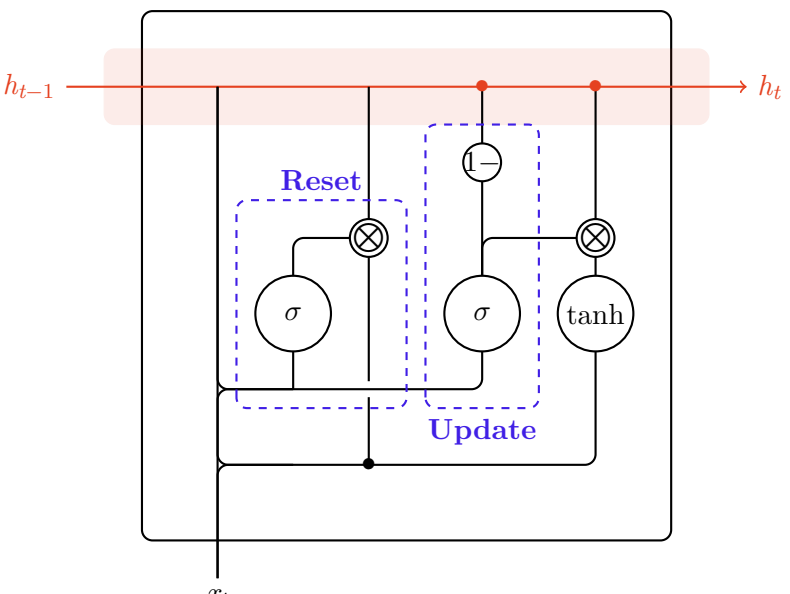


Figure – Cellule GRU

En résumé

Cellules particulières

Problèmes de la couche de Batch Normalization

La couche de Batch Normalization [Ioffe and Szegedy, 2015] estime la moyenne et l'écart-type sur un batch d'observations. Il est difficile de voir comment cela peut se comporter avec un réseau récurrent qui n'a pas forcément une taille de batch fixe. Au lieu de normaliser feature par feature, normalisons observations par observations : c'est la couche Layer Normalization [Ba et al., 2016]

$$\hat{x}_i = \frac{x_i - \mu_l}{\sqrt{\sigma_l^2 + \varepsilon}} \tag{1}$$

$$\mu_l = \frac{1}{d}\sum_{j=1}^{d} x_j \qquad \sigma_l^2 = \frac{1}{d}\sum_{j=1}^{d}(x_i - \mu_l)^2 \qquad \varepsilon > 0$$

Puis, de manière identique à Batch-Normalization pour conserver la puissance de représentation du réseau, l'output de la couche est définie comme :

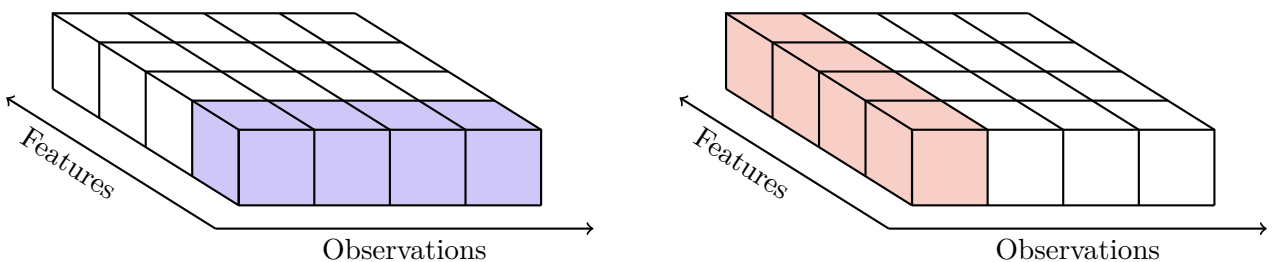
$$z_i = \gamma \hat{x}_i + \beta$$

Avec γ et β des paramètres appris en même temps que les autres paramètres du modèle

Cellules particulières

Comparaison entre Batch et Layer Normalization

Nous avions vu à la séance précédente la couche de Batch Normalization, et on peut comparer le fonctionnement avec la couche de Layer Normalization.



(a) Batch Normalization **(b)** Layer Normalization

Figure – Pour un dataset donné, comparaison de la normalisation entre les deux méthodes décrites

En résumé

Retour de la convolution

Convolution en une dimension

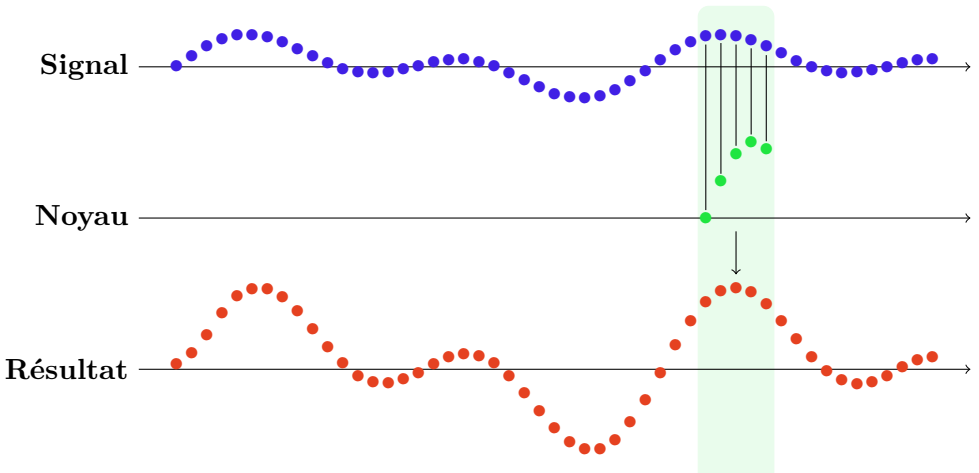


Figure – Résultat de la convolution d'un signal par un noyau

Retour de la convolution

Convolution causale

Une couche de convolution en une dimension utilise à la fois le passé et le futur puisqu'elle *cherche* des schémas spatiaux indépendamment du temps. Pour l'appliquer à une série temporelle, où le temps compte, nous utilisons une **convolution causale**.

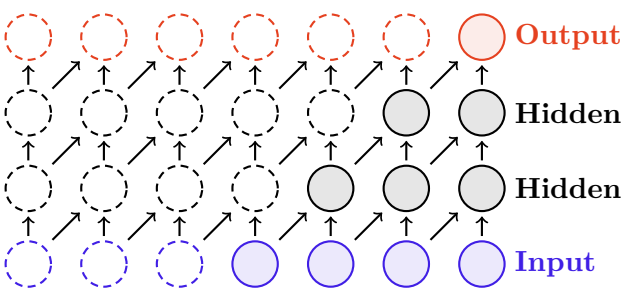


Figure – Schéma d'une convolution causale

Notons que plus on augmente la profondeur du réseau plus on peut exploiter des informations du passé.

Retour de la convolution

Architecture WaveNet

Deepmind présente Wavenet [Van Den Oord et al., 2016] un modèle initialement pour générer de l'audio à partir du texte. L'architecture introduit la couche de **dilated convolution**. Elle permet d'exploiter des informations du passé lointain sans pour autant augmenter fortement la profondeur du réseau.
L'architecture exploite également des *skip-connections*, *residuals-connections* comme dans un ResNet ainsi que des fonctions d'activations par porte comme dans un LSTM.

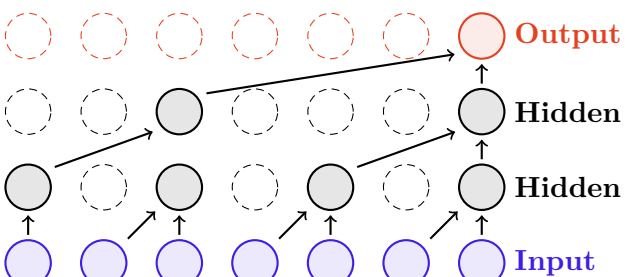


Figure – Schéma d'une *Dilated convolution*

En résumé

Bibliographie I

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. (2023).
Palm 2 technical report.
arXiv preprint arXiv :2305.10403.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016).
Layer normalization.
arXiv preprint arXiv :1607.06450.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning phrase representations using rnn encoder-decoder for statistical machine translation.
arXiv preprint arXiv :1406.1078.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022).
Palm : Scaling language modeling with pathways.
arXiv preprint arXiv :2204.02311.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016).
Lstm : A search space odyssey.
IEEE transactions on neural networks and learning systems.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation.

Bibliographie II

Ioffe, S. and Szegedy, C. (2015).
Batch normalization : Accelerating deep network training by reducing internal covariate shift.
International conference on machine learning.

Sak, H., Senior, A., and Beaufays, F. (2014).
Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition.
arXiv preprint arXiv :1402.1128.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a).
Llama : Open and efficient foundation language models.
arXiv preprint arXiv :2302.13971.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b).
Llama 2 : Open foundation and fine-tuned chat models.
arXiv preprint arXiv :2307.09288.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al. (2016).
Wavenet : A generative model for raw audio.
arXiv preprint arXiv :1609.03499.

Bibliographie III


Zaremba, W., Sutskever, I., and Vinyals, O. (2014).
Recurrent neural network regularization.
arXiv preprint arXiv :1409.2329.


Zhang, B. and Sennrich, R. (2019).
Root mean square layer normalization.
Advances in Neural Information Processing Systems, 32.

Annexe : RMSNorm

Simplifions la normalisation

La couche Layer Normalization induit un surplus de calcul qui peut devenir coûteux quand on entraîne des LLM. [Zhang and Sennrich, 2019] propose une alternative à Layer Normalization baptisé RMSNorm dans l'article.

L'hypothèse de départ de l'article est que la stabilisation offerte par les techniques de normalisation ne vient pas du retrait de la moyenne, mais de la mise à l'échelle. Ainsi, il propose de simplifier LayerNorm en supprimant la dépendance en la moyenne et en modifiant l'input comme :

$$\hat{x}_i = \frac{x_i}{\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_i^2}} \tag{2}$$

En comparant aux équations de la BatchNormalization et la LayerNormalization on observe la simplification croissante de la normalisation. Cependant, les performances sont comparables ! Ainsi, les modèles LLaMa de Meta [Touvron et al., 2023a, Touvron et al., 2023b] et PaLM de Google [Chowdhery et al., 2022, Anil et al., 2023] utilise RMSNorm.