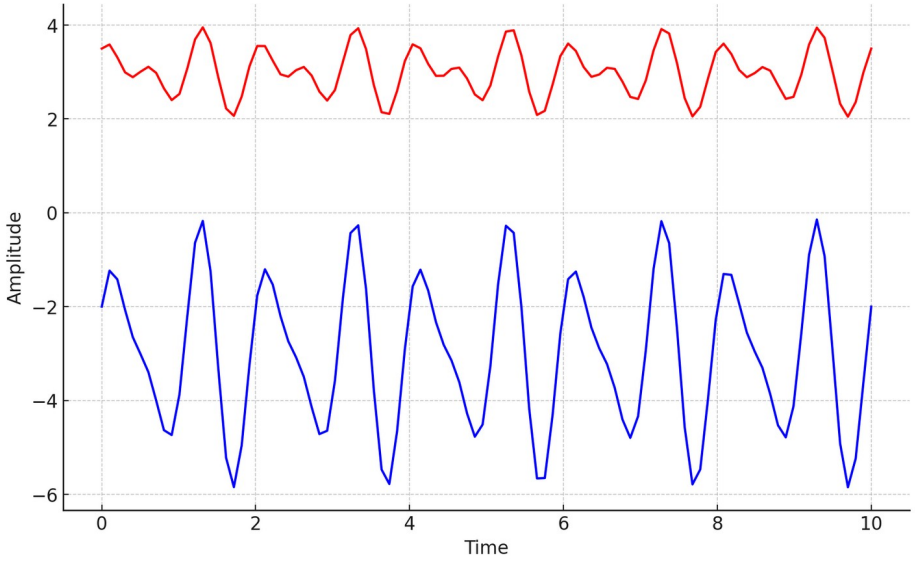
Présentation des Modèles Mamba et Jamba

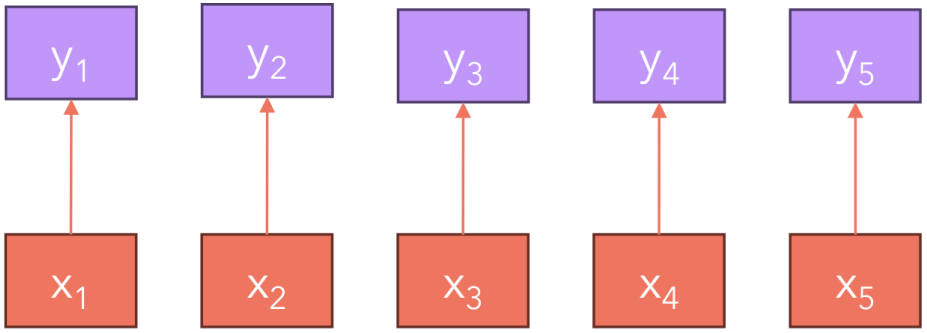
Robert Maria

Strasbourg
29.06.2024

Qu'est-ce que la modélisation de séquences ?



Domaine continu



Domaine discret

Pourquoi Mamba ?

Pour surmonter les limitations des **transformers !!!**

Pourquoi Mamba ?

Pour surmonter les limitations des **transformers !!!**

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu[*1] and Tri Dao[*2]

[1]Machine Learning Department, Carnegie Mellon University

[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

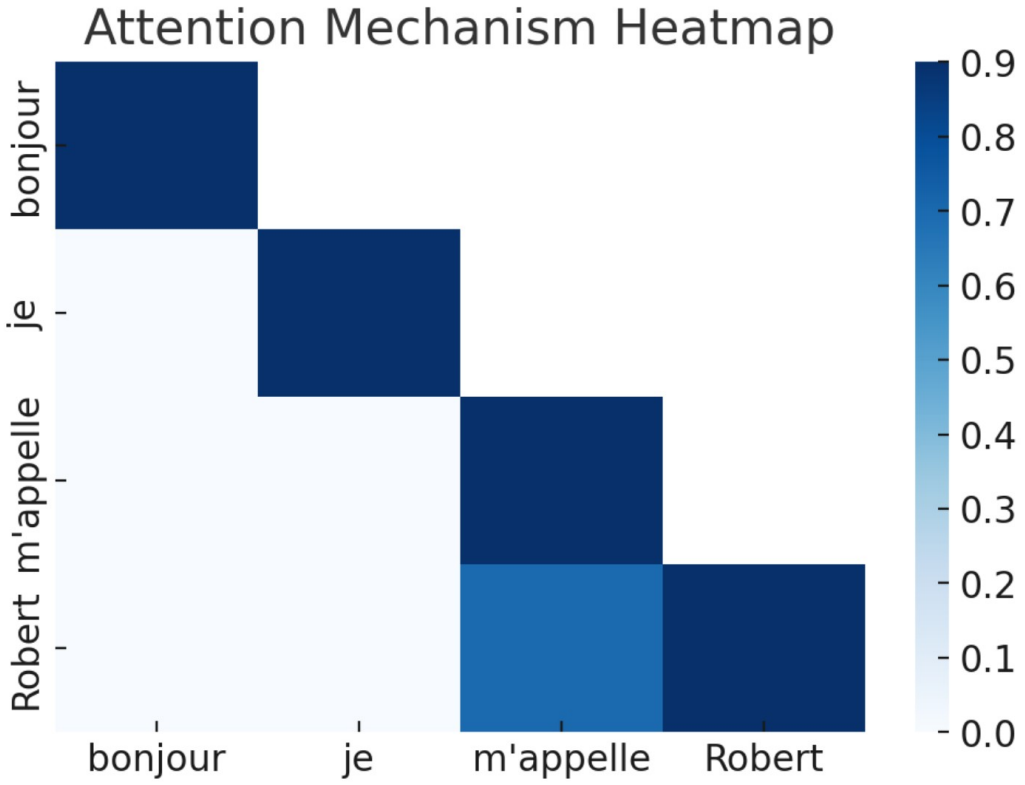
Quoi ?? Les transformers ont des problèmes ?

Quoi ?? Les transformers ont des problèmes ?

Entraînement	Inférence
Rapide (parallélisable)	Lente (quadratique avec la longueur de l'input, ou linéaire si nous mettons en cache les KV)

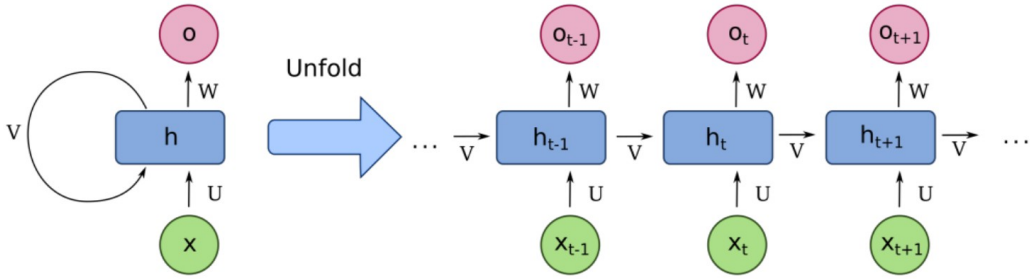
Quoi ?? Les transformers ont des problèmes ?

Entraînement	Inférence
Rapide (parallélisable)	Lente (quadratique avec la longueur de l'input, ou linéaire si nous mettons en cache les KV)



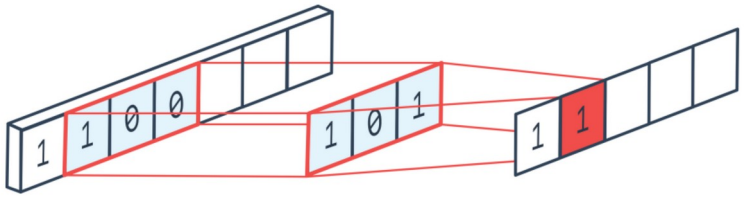
L^2 computations

Quoi ?? Les transformers ont des problèmes ?



source

Réseaux de Neurones Récurrents (RNN)



source

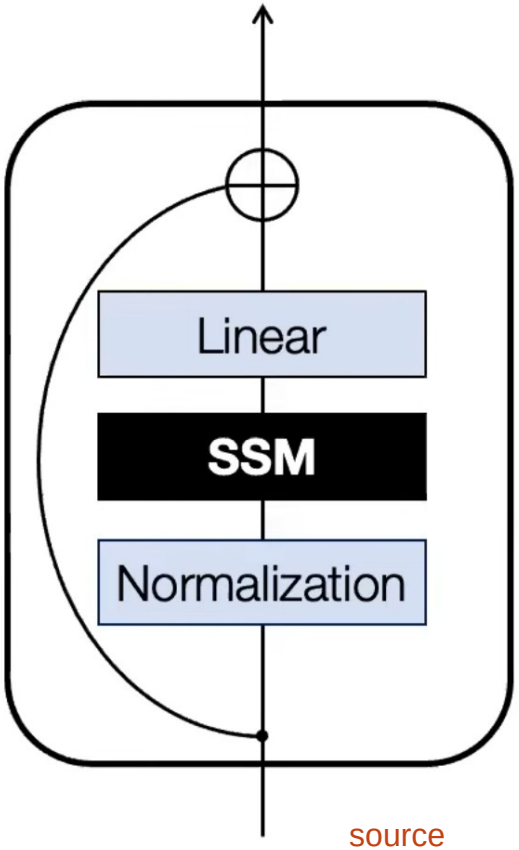
Réseaux de Neurones Convolutionnels (CNN)

Chaque état caché h est l'agrégation de tous les états cachés précédents et est généralement compressée

Entraînement	Inférence
Lente (non parallélisable) Wanishing / exploding gradients	Rapide Infinite context (en théorie)

Entraînement	Inférence
Rapide (parallélisable)	Rapide Contexte finie (en fonction de la taille du kernel)

State Space Models



source

Les SSMs sont des modèles probabilistiques.

Nous pouvons les ajouter en tant que module dans un réseau neuronal

State Space Models

permet de mapper un signal d'entrée x(t) à un signal de sortie y(t) par l'intermédiaire d'une représentation h(t) :

$$x' = \boldsymbol{A}x + \boldsymbol{B}u$$
$$y = \boldsymbol{C}x + \boldsymbol{D}u$$

A ϵ N×N matrice; B ϵ N vecteur colonne

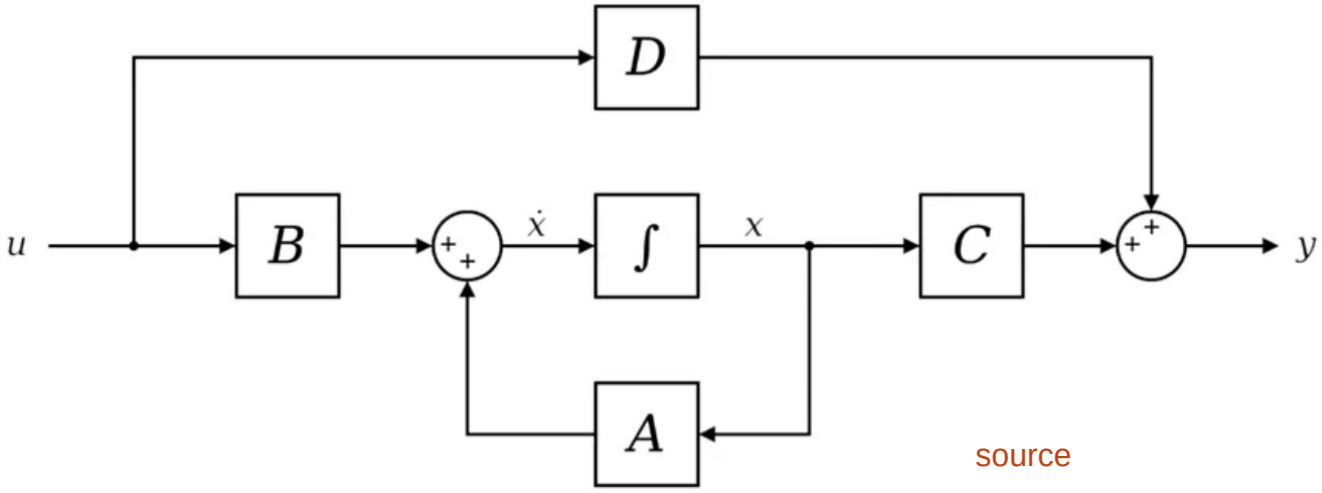
State Space Models

permet de mapper un signal d'entrée x(t) à un signal de sortie y(t) par l'intermédiaire d'une représentation h(t) :

$$x' = \boldsymbol{A}x + \boldsymbol{B}u$$

$$y = \boldsymbol{C}x + \boldsymbol{D}u$$

A ϵ N×N matrice; B ϵ N vecteur colonne



source

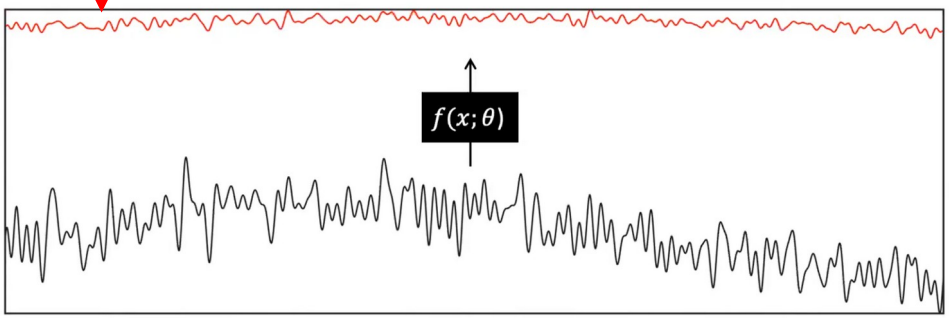
State Space Models

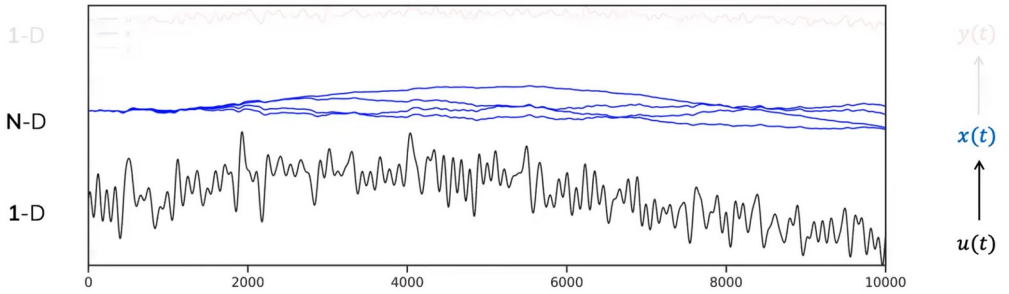
permet de mapper un signal d'entrée x(t) à un signal de sortie y(t) par l'intermédiaire d'une représentation h(t) :

$$x' = Ax + Bu$$

$$y = Cx + Du$$

A ϵ N×N matrice; B ϵ N vecteur colonne





Discretization

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t)$$
$$y(t) = Ch(t) + Dx(t)$$

$$\lim_{\Delta \to 0} \frac{h(t+\Delta) - h(t)}{\Delta} = h'(t)$$

$$\frac{b(t+\Delta) - b(t)}{\Delta} \approx b'(t)$$

$$h(t+\Delta) \approx \Delta h'(t) + h(t)$$

la méthode d'Euler

Discretization

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t)$$

$$h(t+\Delta) \approx \Delta h'(t) + h(t)$$

la méthode d'Euler

$$\begin{aligned} h(t+\Delta) &\approx \Delta(Ah(t) + Bx(t)) + h(t) \\ &= \Delta Ah(t) + \Delta Bx(t) + h(t) \\ &= (I + \Delta A)h(t) + \Delta Bx(t) \\ &= \bar{A}h(t) + \bar{B}x(t) \end{aligned}$$

Discretization

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t) \quad (1a) \qquad h_t = \overline{\boldsymbol{A}}h_{t-1} + \overline{\boldsymbol{B}}x_t$$

$$y(t) = \boldsymbol{C}h(t) \quad (1b) \qquad y_t = \boldsymbol{C}h_t$$

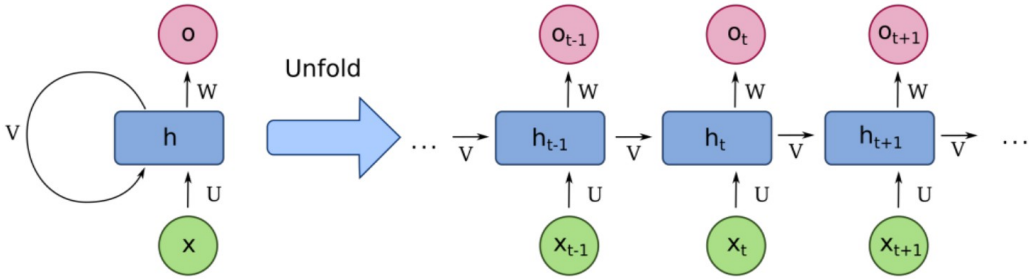
zero-order hold

$$\overline{\boldsymbol{A}} = \exp(\Delta \boldsymbol{A}) \qquad \overline{\boldsymbol{B}} = (\Delta \boldsymbol{A})^{-1}(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I}) \cdot \Delta \boldsymbol{B}$$

Le paramètre Δ est appris avec la descente de gradient

Calcul récurrent

$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$
$$y_t = C h_t$$



source

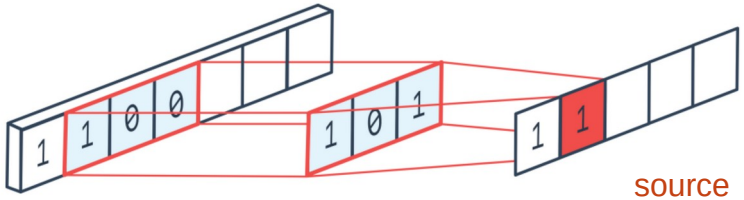
$$h_0 = \bar{B} x_0$$
$$y_0 = C h_0 = C \bar{B} x_0$$

$$h_1 = \bar{A} h_0 + \bar{B} x_1 = \bar{A} \bar{B} x_0 + \bar{B} x_1$$
$$y_1 = C h_1 = C(\bar{A} \bar{B} x_0 + \bar{B} x_1) = C \bar{A} \bar{B} x_0 + C \bar{B} x_1$$

$$h_2 = \bar{A} h_1 + \bar{B} x_2 = \bar{A}^2 \bar{B} x_0 + \bar{B} x_1 + \bar{B} x_2$$
$$y_2 = C h_2 = C \bar{A}^2 \bar{B} x_0 + C \bar{A} \bar{B} x_1 + \bar{B} x_2$$

$$y_k = C \bar{A}^k \bar{B} x_0 + C \bar{A}^{k-1} \bar{B} x_1 + \ldots + C \bar{A} \bar{B} x_{k-1} + C \bar{B} x_k$$

Calcul convolutif



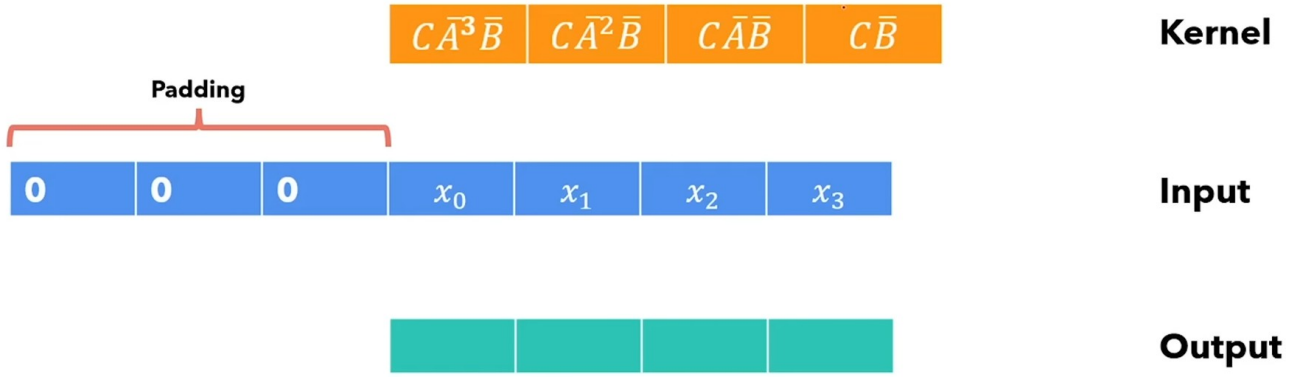
source

$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \ldots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

Construire un kernel pour calculer cela ?

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{b}, \ldots, C\bar{A}^k\bar{b})$$

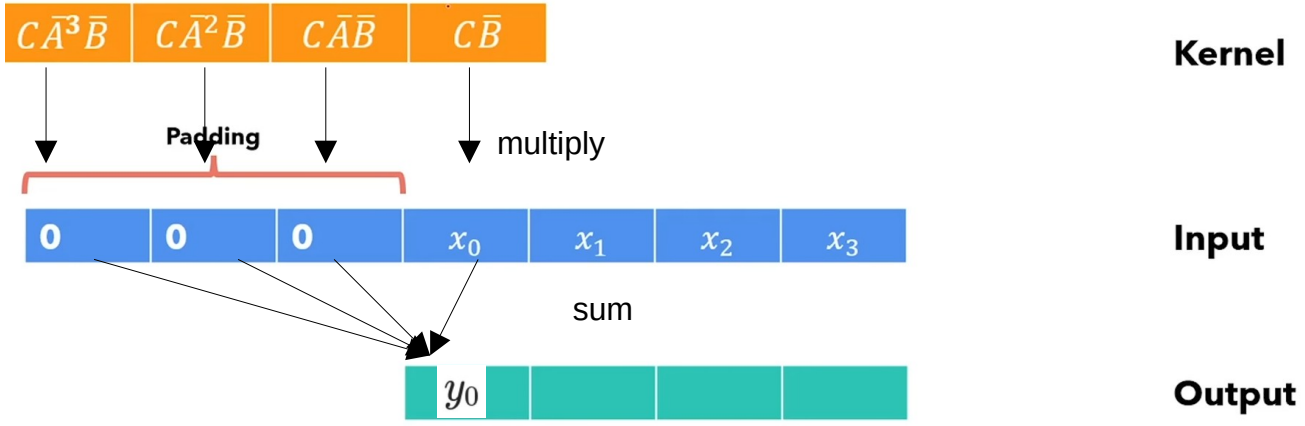
$$y = x * \bar{K}$$



Calcul convolutif

$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \ldots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{b}, \ldots, C\bar{A}^k\bar{b}) \qquad y = x * \bar{K}$$

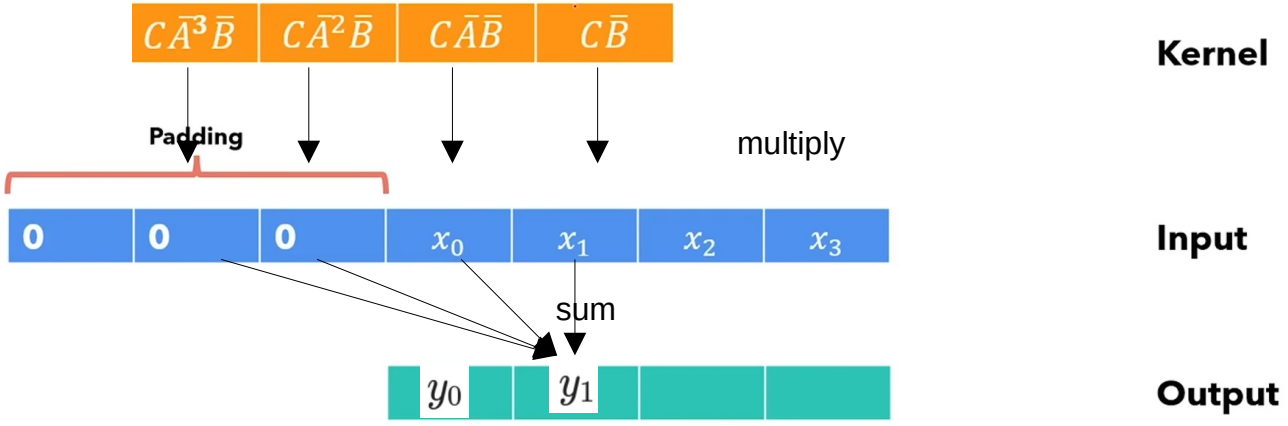


$$y_0 = C\bar{B}x_0$$

Calcul convolutif

$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \ldots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{b}, \ldots, C\bar{A}^k\bar{b}) \qquad y = x * \bar{K}$$

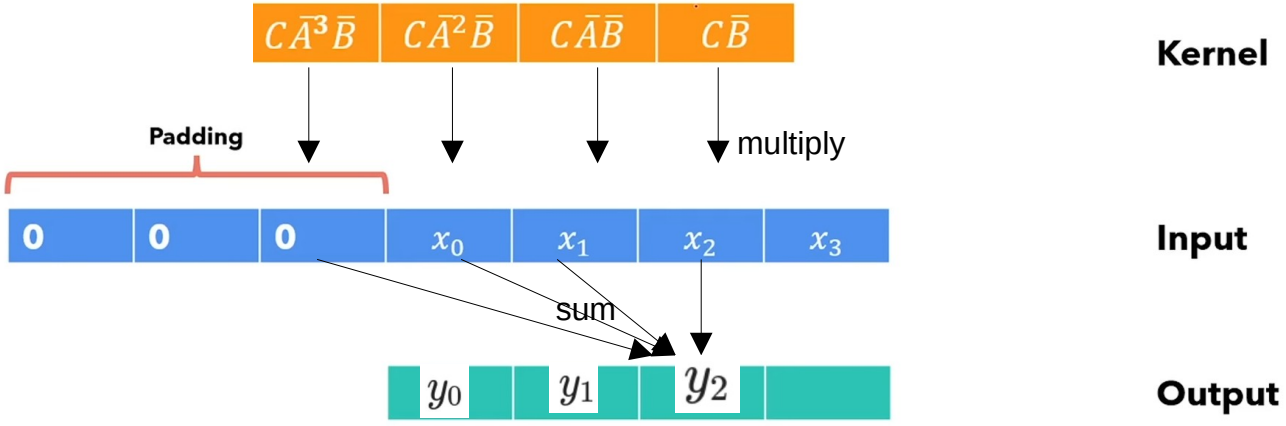


$$y_1 = C\bar{A}\bar{B}x_0 + C\bar{B}x_1$$

Calcul convolutif

$$y_k = C\bar{A}^k\bar{B}x_0 + C\bar{A}^{k-1}\bar{B}x_1 + \ldots + C\bar{A}\bar{B}x_{k-1} + C\bar{B}x_k$$

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{b}, \ldots, C\bar{A}^k\bar{b}) \qquad y = x * \bar{K}$$

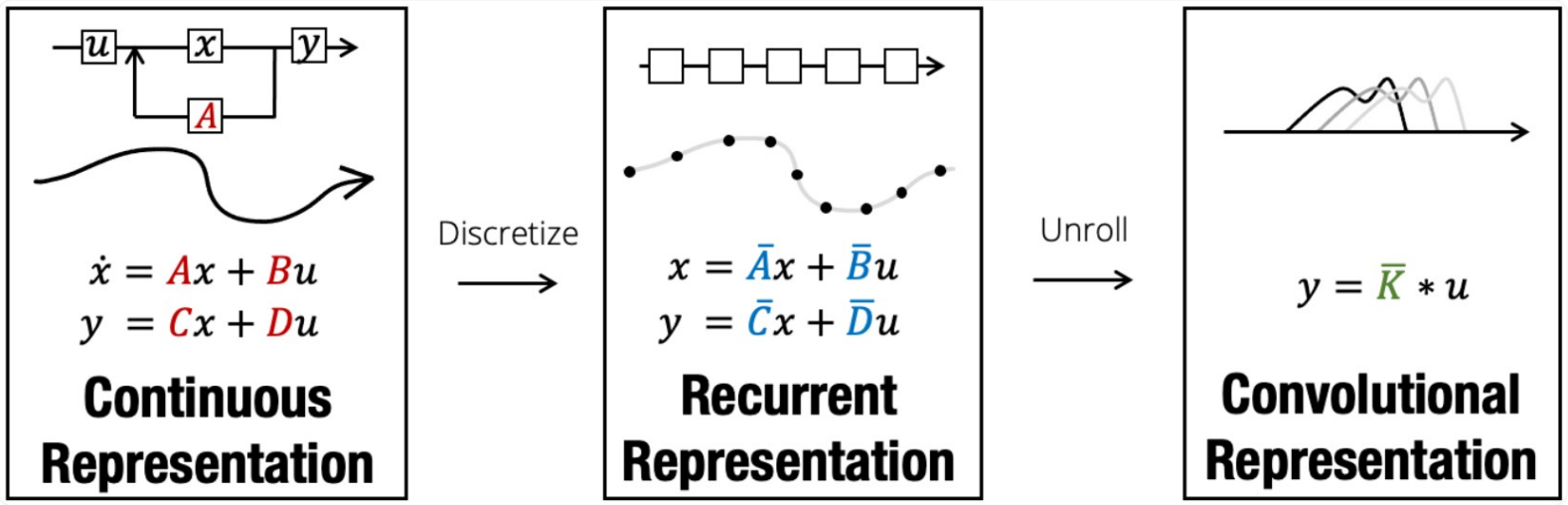


$$y_2 = C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + \bar{B}x_2$$

Avantages

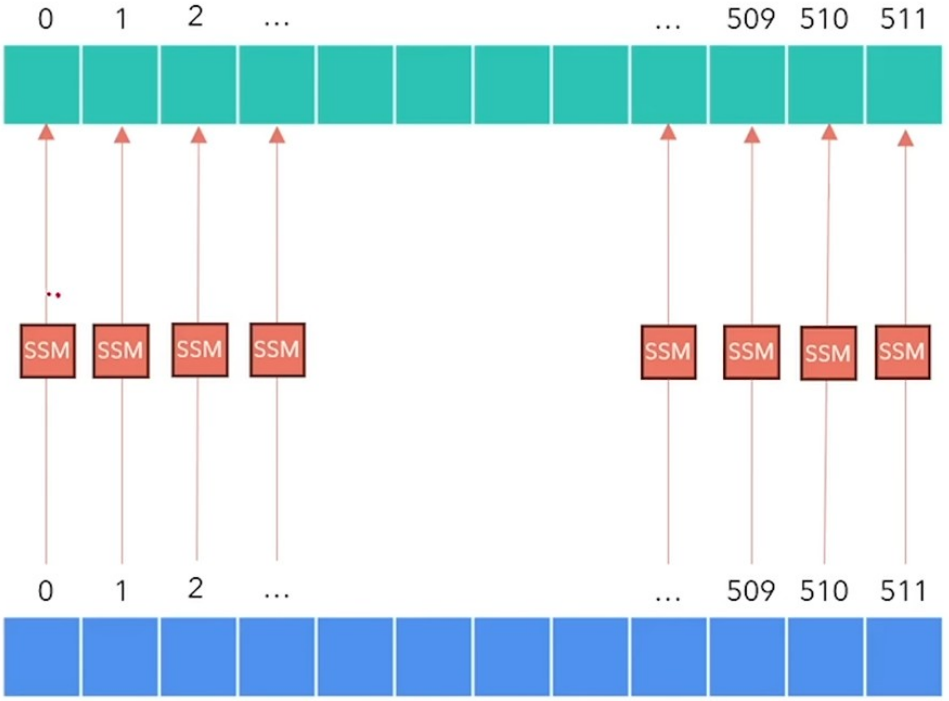
Calcul par convolution pour effectuer l'entraînement

Calcul récurrent pour effectuer l'inférence, un token à la fois, en utilisant une quantité constante de calcul pour chaque token.

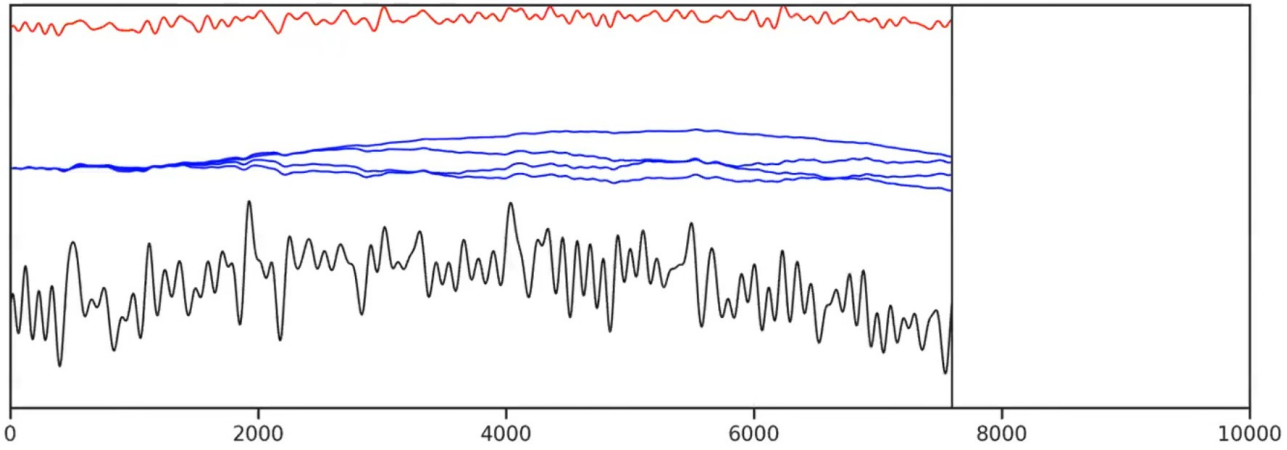


source

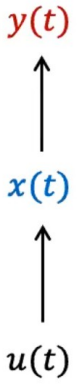
State Space Models - avec 512 dimensions



source



source



Structured State Space (S4)

S4 = SSM + HIPPO + Structured Matrices

formules spéciales pour les matrices A et B, sont appelées matrices HIPPO

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 3 & 3 & 0 \\ 1 & 3 & 5 & 4 \end{bmatrix}$$

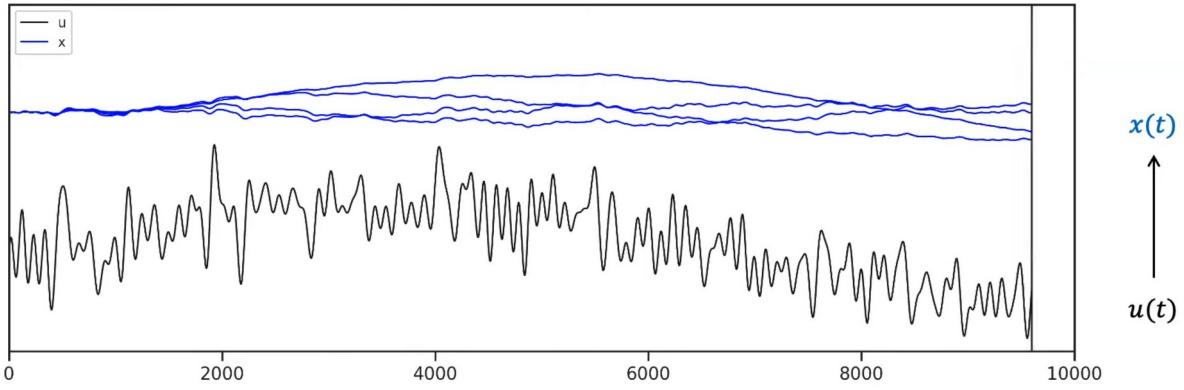
$$x' = \boldsymbol{A}x + \boldsymbol{B}u$$

HiPPO

$$y = \boldsymbol{C}x + \boldsymbol{D}u$$

source

HIPPO est une correspondance entre u(t) et x(t)

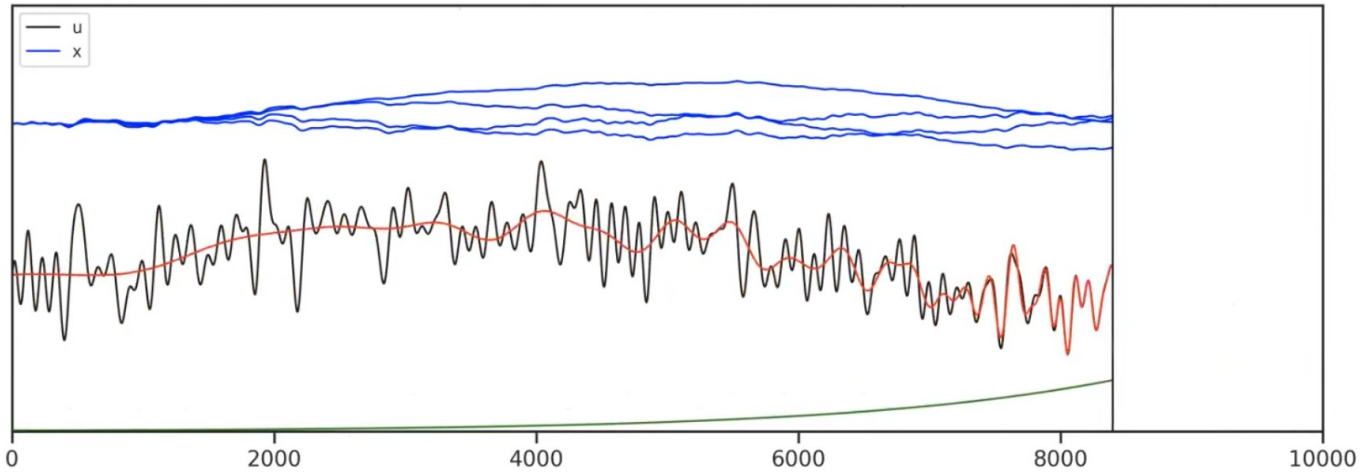


High-Order Polynomial Projection Operator

1) observer la séquence d'entrée: u1, u2, u3, ...

2) pour chaque étape temporelle, calculer les : x1, x2, x3, ...

3) encoder l'historique de toutes les entrées, c'est possible ?



source

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$

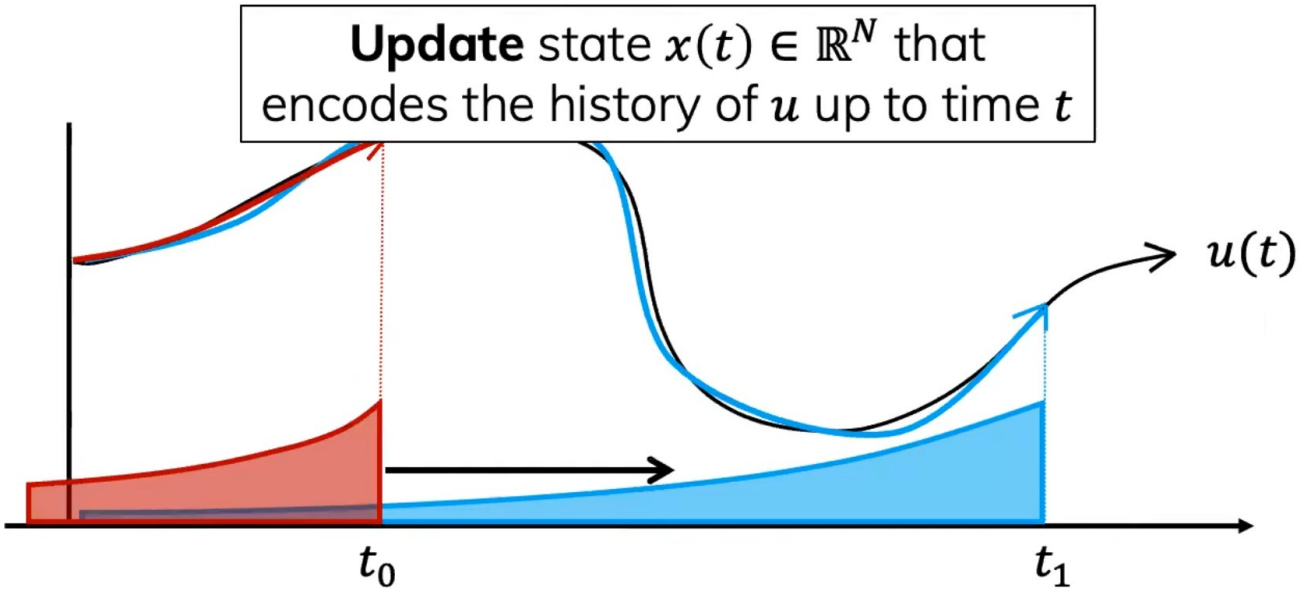
operateur HIPPO

$$\boldsymbol{A}_{nk} = \begin{cases} 0 & n < k \\ n+1 & n = k \\ 2n+1 & n > k \end{cases}$$

matrice HIPPO

source

High-Order Polynomial Projection Operator

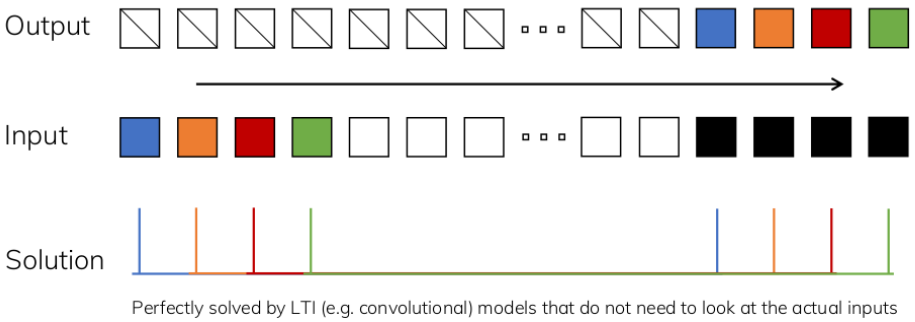


$$x(t_0) = \begin{bmatrix} 0.1 \\ -1.1 \\ 3.7 \\ 2.5 \end{bmatrix} \xrightarrow{\text{update}} x(t_1) = \begin{bmatrix} 1.5 \\ 2.9 \\ -0.3 \\ 2.0 \end{bmatrix}$$

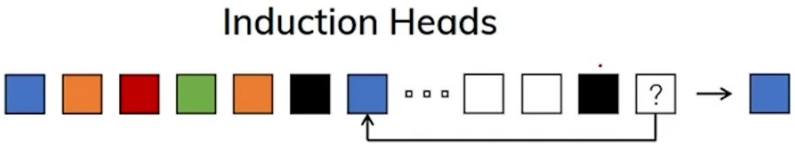
source

Motivation

Copie

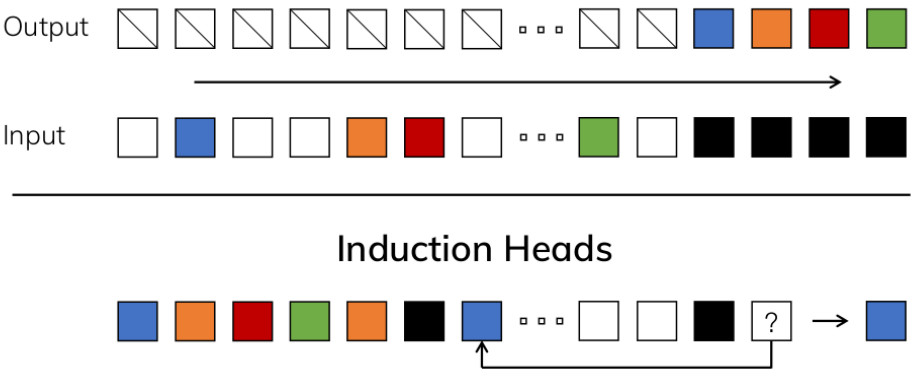


réécrire l'entrée un token à la fois, mais avec un décalage temporel



source

Copie sélective



étant donné un post sur Reddit, supprimer tous les gros mots

ne peut pas être réalisé par des SSMs normaux

Améliorer les SSMs avec la Sélection

Algorithm 1 SSM (S4)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

1: $\boldsymbol{A} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
3: $\boldsymbol{C} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
4: $\Delta : (\mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter})$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
7: **return** y

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

1: $\boldsymbol{A} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_B(x)$
3: $\boldsymbol{C} : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_C(x)$
4: $\Delta : (\mathsf{B}, \mathsf{L}, \mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathsf{B}, \mathsf{L}, \mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
7: **return** y

B: Batch Size
L: Sequence Length
D: Size of the input vector (equivalent to *d_model* in the Transformer)
N: Size of the hidden state h.

source

Problème: tous les modèles SSM précédents doivent être invariants au temps et à l'entrée afin d'être efficaces sur le plan computationnel

L'opération de balayage (scan)

Prefiz-sum array: est un tableau calculé séquentiellement, la valeur à chaque position indique la somme des positions précédentes

Prefix-sum	3	5	7	10	14	16
	3	8	15	25	39	51

la formulation récurrente des SSM peut également être considérée comme une opération de balayage dans laquelle chaque état est la somme de l'état précédent et de l'entrée actuelle

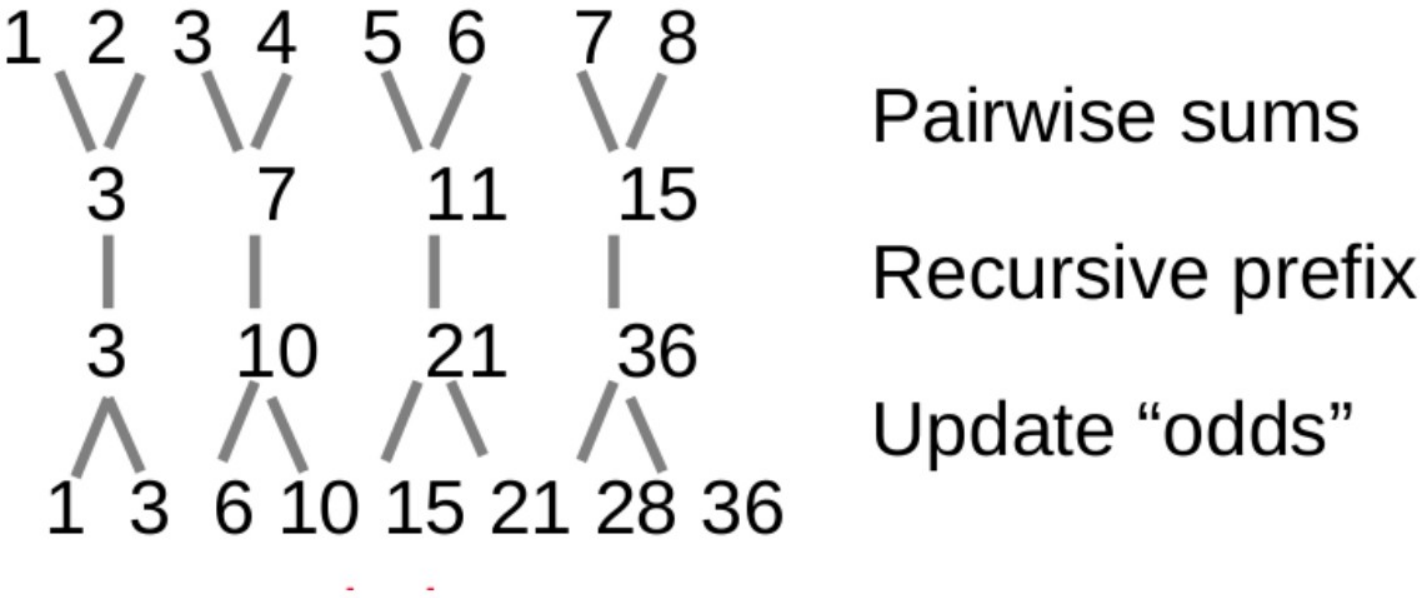
input	x_0	x_1	x_2	x_3	x_4	x_5
scan output	h_0	h_1	h_2	h_3	h_4	h_5

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$

$$y_t = Ch_t$$

Le balayage parallèle (parallel scan)

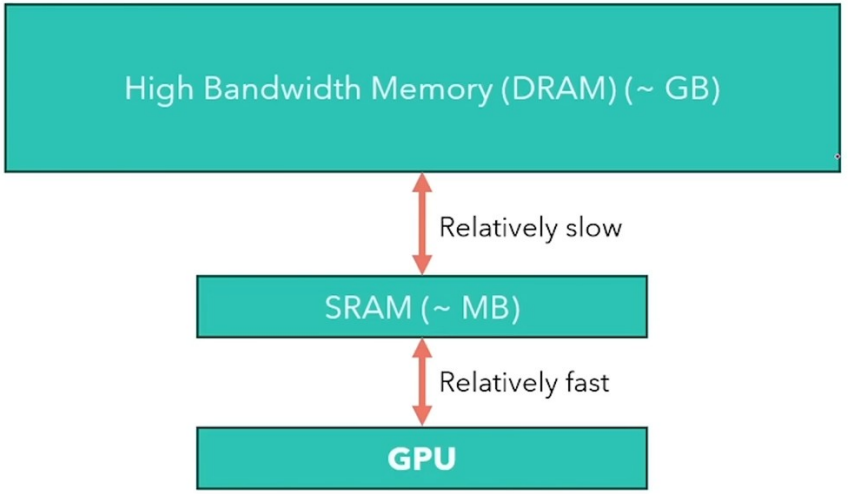
si les opérations sont associatives, nous pouvons paralléliser l'opération de balayage



source

Kernel fusion

créer un CUDA kernel custom pour effectuer plusieurs opérations dans la SRAM et éviter de copier des données intermédiaires dans la HBM



L'architecture de MAMBA

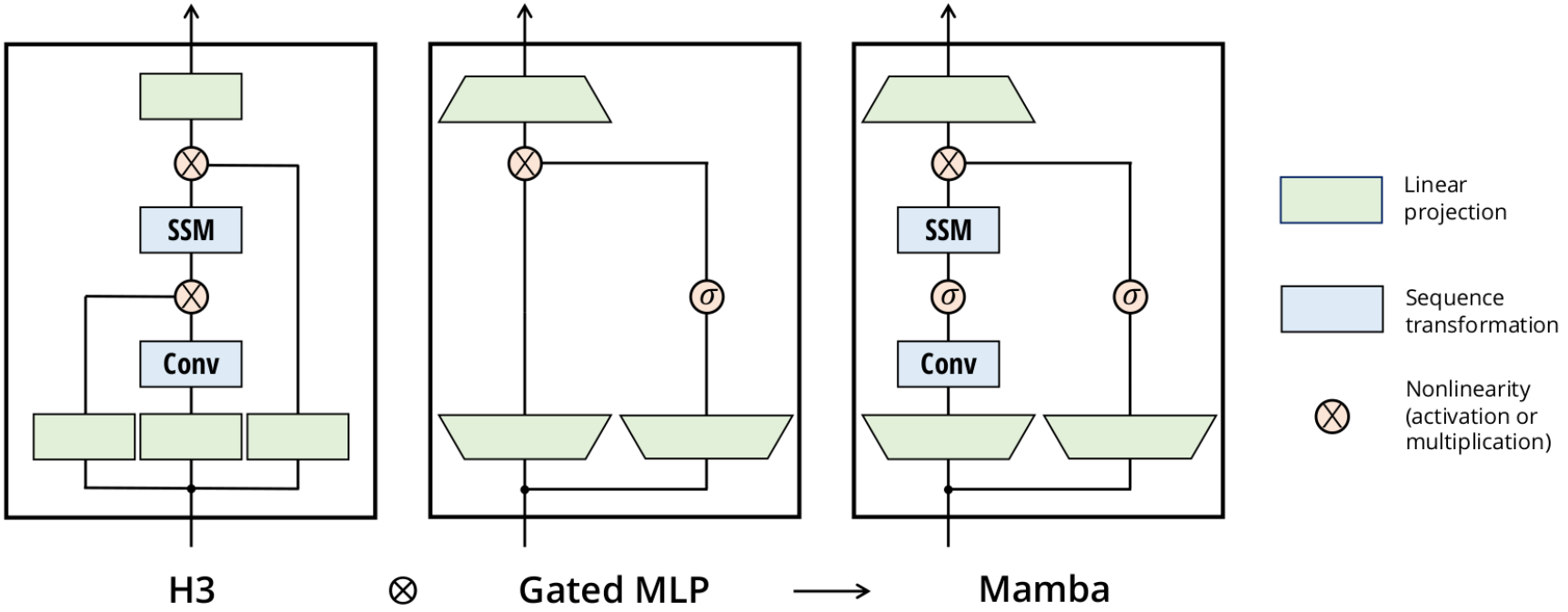


Figure 3: (**Architecture**.) Our simplified block design combines the H3 block, which is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch. For σ we use the SiLU / Swish activation (Hendrycks and Gimpel 2016; Ramachandran, Zoph, and Quoc V Le 2017).

Résultats

Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

Table 1: (**Selective Copying**.) Accuracy for combinations of architectures and inner sequence layers.

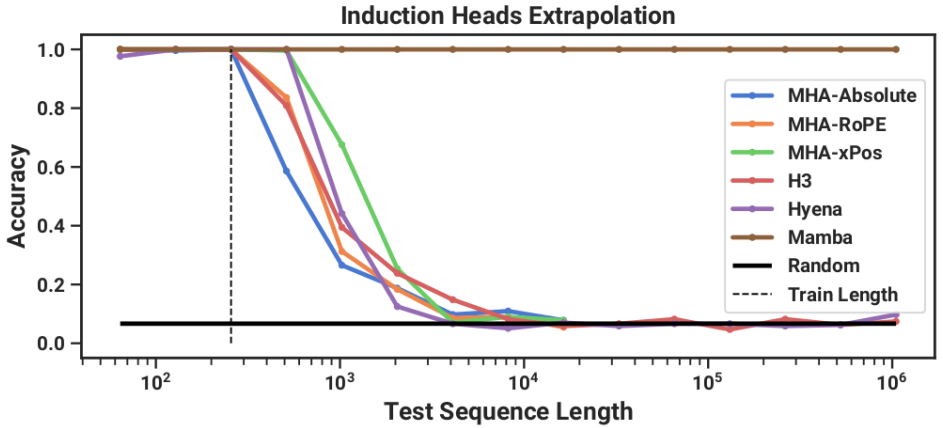


Table 2: (**Induction Heads**.) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

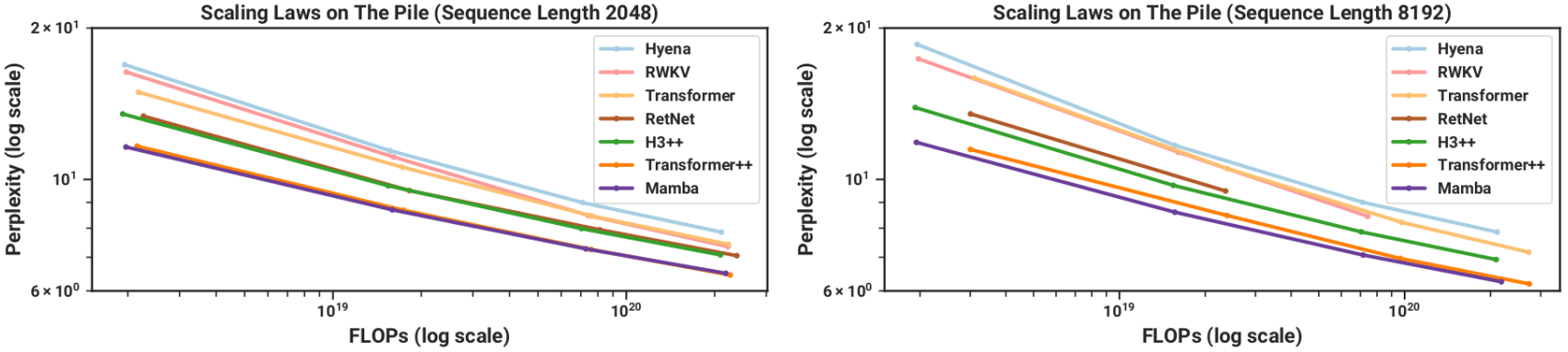


Figure 4: (**Scaling Laws**.) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length grows.