HiPPO et S4 pour la modélisation de séquences avec dépendances à (très) longue portée

une unification des CNN et des RNN et une alternative aux Transformers

Séminaire TALia du 09/09/2022

1

arXiv > cs > arXiv:2008.07669

Computer Science > Machine Learning

[Submitted on 17 Aug 2020 (v1), last revised 23 Oct 2020 (this version, v2)]

HiPPO: Recurrent Memory with Optimal Polynomial Projections

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, Christopher Re

2

arXiv > cs > arXiv:2110.13985

Computer Science > Machine Learning

[Submitted on 26 Oct 2021]

Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, Christopher Ré

3

arXiv > cs > arXiv:2111.00396

Computer Science > Machine Learning

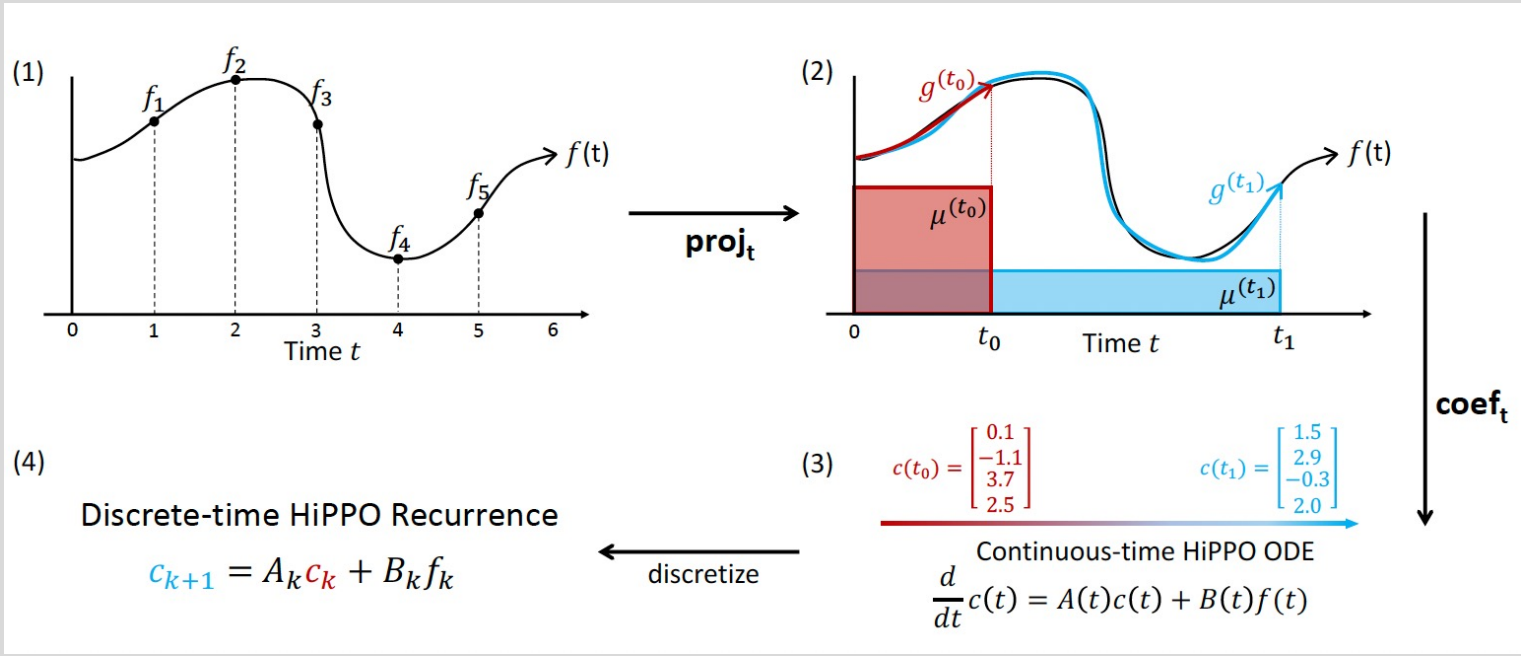
[Submitted on 31 Oct 2021 (v1), last revised 5 Aug 2022 (this version, v3)]

Efficiently Modeling Long Sequences with Structured State Spaces

Albert Gu, Karan Goel, Christopher Ré

Plan

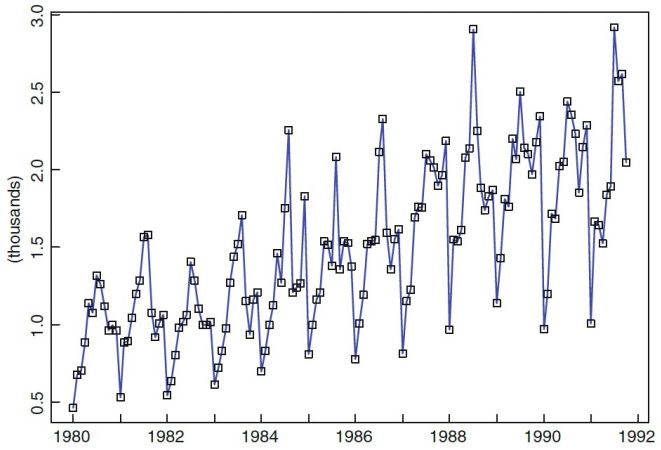
1. Le contexte historique
2. S4 relève le challenge des dépendances à longue portée
3. Les fondations théoriques de S4 avec HiPPO et SSM
4. Les 3 vues de S4 :
   CTM, RNN, CNN
5. Questions ouvertes pour le TAL



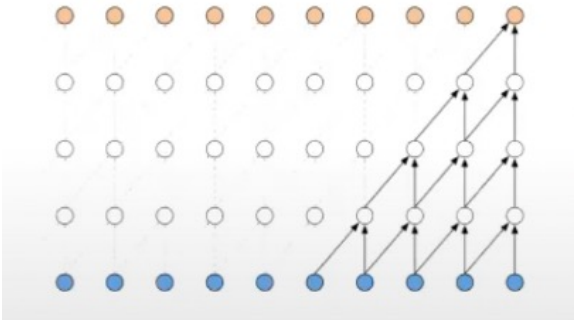
Les modèles de séquences – une longue histoire !

Modèles de séries temporelles ARMA

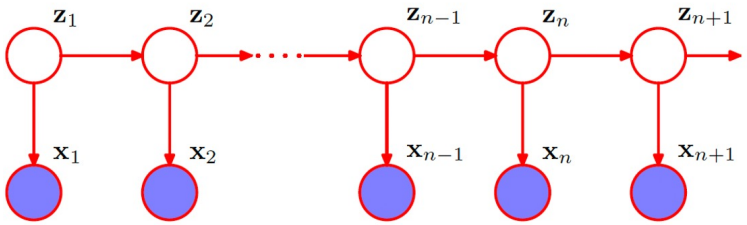
$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$



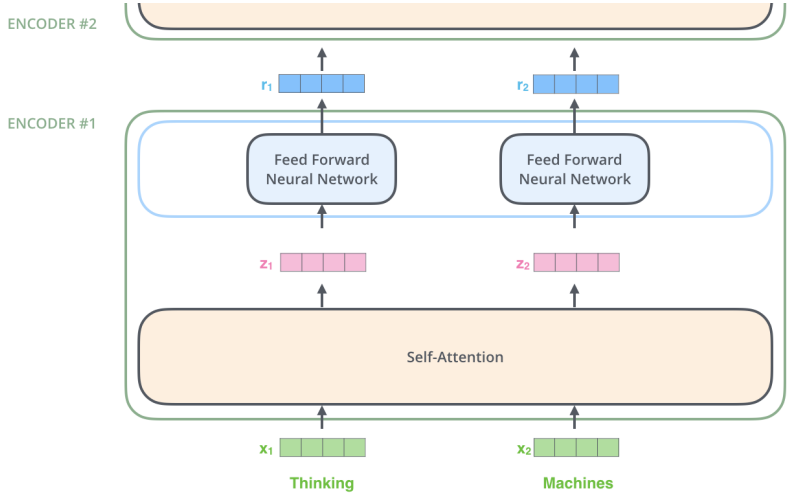
1D CNN



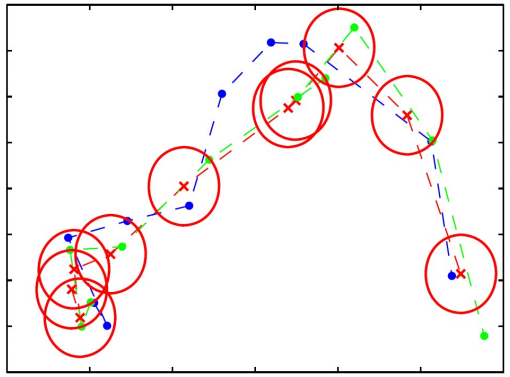
Hidden Markov Models (HMM)
Linear Dynamical Systems (LDS)



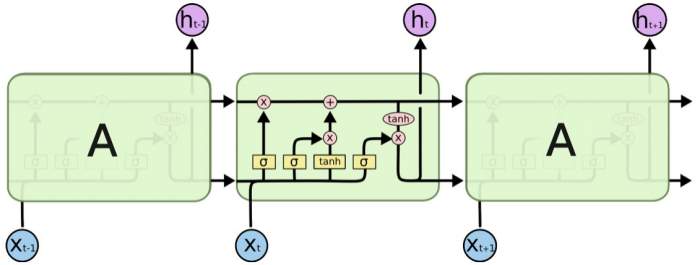
Transformers



SSM, filtres de Kalman



RNN, LSTM, GRU, mécanisme d'attention



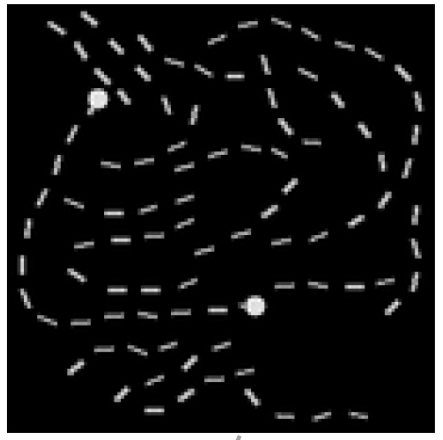
S4 relève le challenge des dépendances à (très) longue portée

S4 versus Transformers les plus performants à l'entraînement

	Length 1024		Length 4096	
	Speed	Mem.	Speed	Mem.
Transformer	1×	1×	1×	1×
Performer	1.23×	0.43×	3.79×	0.086×
Linear Trans.	**1.58×**	**0.37×**	**5.35×**	**0.067×**
S4	**1.58×**	0.43×	5.19×	0.091×

S4 pour la reconnaissance de la parole

	MFCC	Raw	0.5×
Transformer	90.75	✗	✗
Performer	80.85	30.77	30.68
ODE-RNN	65.9	✗	✗
NRDE	89.8	16.49	15.12
ExpRNN	82.13	11.6	10.8
LipschitzRNN	88.38	✗	✗
CKConv	**95.3**	71.66	65.96
WaveGAN-D	✗	96.25	✗
LSSL	93.58	✗	✗
S4	93.96	**98.32**	**96.30**



S4 comme modèle de langue

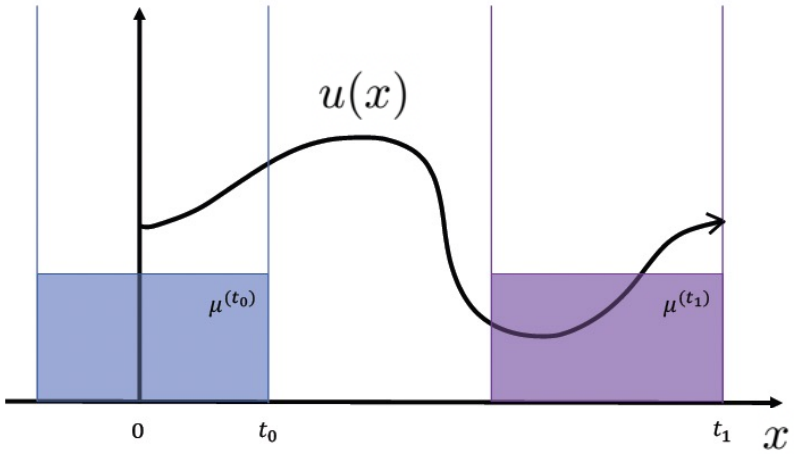
Model	Params	Test ppl.	Tokens / sec
Transformer	247M	**20.51**	0.8K (1×)
GLU CNN	229M	37.2	-
AWD-QRNN	151M	33.0	-
LSTM + Hebb.	-	29.2	-
TrellisNet	180M	29.19	-
Dynamic Conv.	255M	25.0	-
TaLK Conv.	240M	23.3	-
S4	249M	**21.28**	**48K (60×)**

S4 sur les 6 tâches du benchmark Long Range Arena

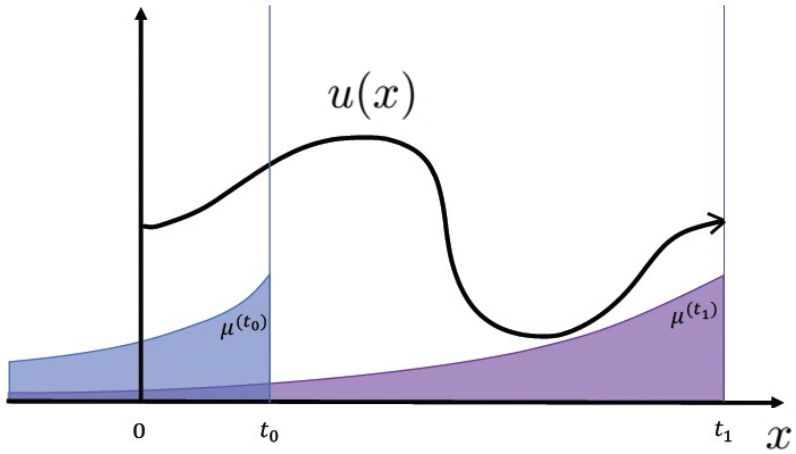
Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
BigBird	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans.	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nyströmformer	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256	37.25	64.57	79.29	47.38	77.72	✗	59.37
S4	**58.35**	**76.02**	**87.09**	**87.26**	**86.05**	**88.10**	**80.48**

HiPPO : mémorisation optimale et incrémentale de l'histoire

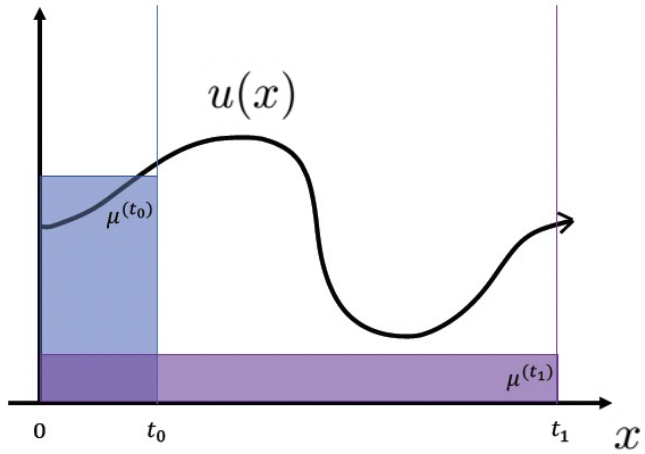
(1) choisir une pondération des évènements passés



Translated Legendre (LegT)



Translated Laguerre (LagT)



Scaled Legendre (LegS)

HiPPO : mémorisation optimale et incrémentale de l'histoire

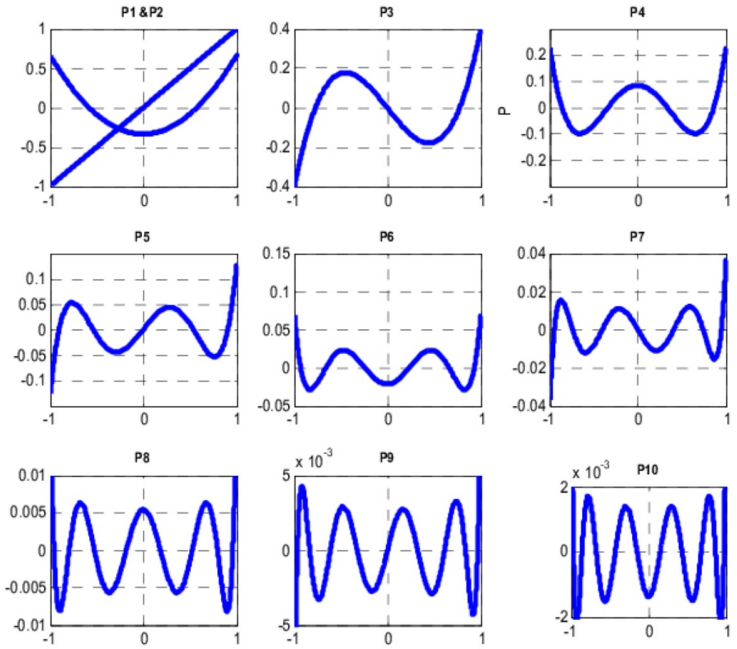
(2) choisir une base de fonctions $\left\{v_i^{(t)}\right\}_{i\leq N}$ pour approximer l'historique $u_{\leq t} := u(x)|_{x\leq t}$ par une combinaison

$$g^{(t)}(x) := \sum_{i=1}^{N} c_i(t)\, v_i^{(t)}(x)$$

qui minimise $\left\|u_{\leq t} - g^{(t)}\right\|_{L^2(\mu^{(t)})}$.

Pour des polynômes orthogonaux $\left\langle v_i^{(t)}, v_j^{(t)} \right\rangle_{\mu^{(t)}} = \delta_{ij}$ on a

$$c_i(t) = \left\langle v_i^{(t)}, u_{\leq t} \right\rangle_{\mu^{(t)}} = \int_0^t v_i^{(t)}(x)\, u_{\leq t}(x)\, \mathrm{d}\mu^{(t)}(x)$$



HiPPO = **Hi**gher Order **P**olynomial **P**rojection **O**perators

HiPPO : mémorisation optimale et incrémentale de l'histoire

(3) écrire l'ED pour les coefficients $c(t) := (c_1(t), \dots, c_N(t)) \in \mathbb{R}^N$. On peut montrer que pour toute mesure $\mu^{(t)}$ on a une **ED linéaire** :

$$\frac{\mathrm{d}c(t)}{\mathrm{d}t} = \boldsymbol{A}(t)\, c(t) + \boldsymbol{B}(t)\, u(t), \quad \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{B} \in \mathbb{R}^{1\times N}$$

mémoire instantanée compressée du signal $u(t)$ pour un budget N.

signal scalaire

$$\Rightarrow \quad c = \mathrm{hippo}(u)$$

non-trivial car il faut aussi dériver la mesure dans

$$c_i(t) = \int_0^t v_i^{(t)}(x)\, u_{\leq t}(x)\, \mathrm{d}\mu^{(t)}(x)$$

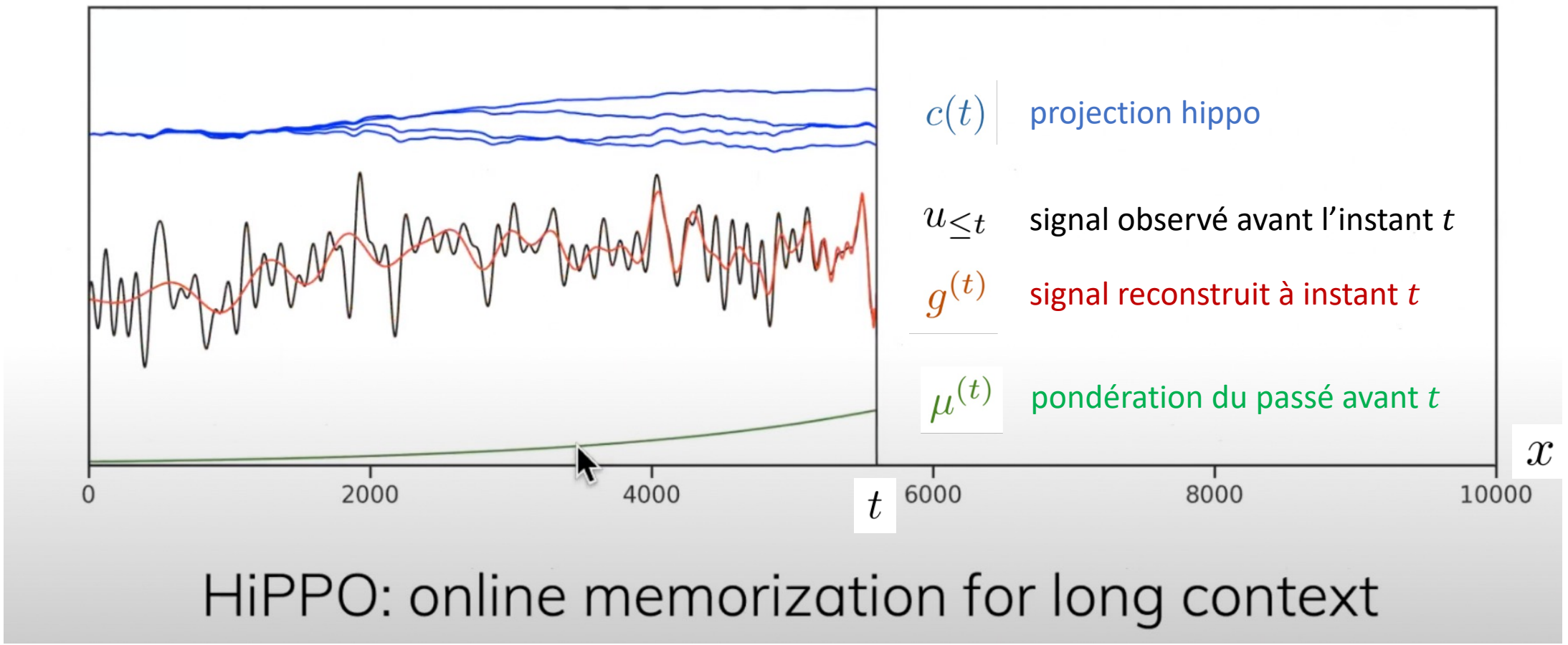
Les matrices $\boldsymbol{A}$ et $\boldsymbol{B}$ sont **explicitement calculables** pour les $\mu^{(t)}$ classiques.

Pour LegS $\quad \mu^{(t)} = \frac{1}{t}\mathbb{I}_{[0,t]} \qquad -t\, A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{si } n > k \\ n+1 & \text{si } n = k \\ 0 & \text{si } n < k \end{cases}, \qquad t\, B_n = (2n+1)^{\frac{1}{2}}$

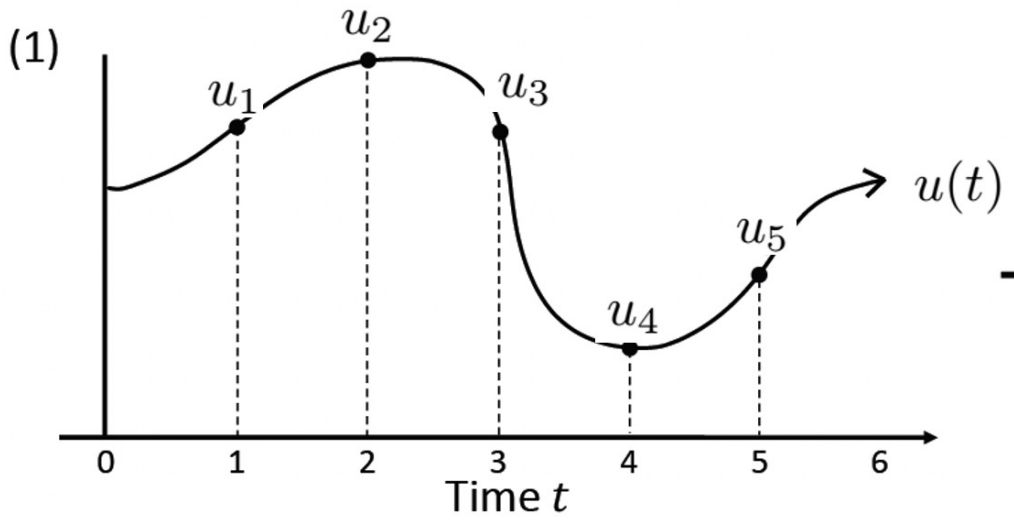
Matrice HiPPO

Pour LegS on a **équivariance sous les changements d'échelle :** $\quad \gamma : t \to \alpha t \qquad \mathrm{hippo}(u \circ \gamma) = \mathrm{hippo}(u) \circ \gamma$

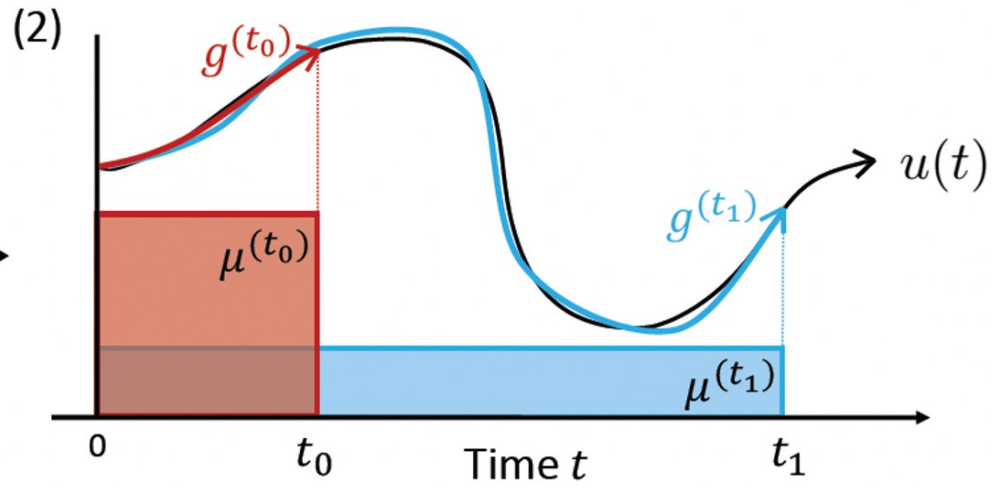
HiPPO : reconstruction à la volée du signal à partir de l'état



HiPPO : une séquence comme discrétisation d'un signal



proj$_t$



coef$_t$

(3)

$$c(t_0) = \begin{bmatrix} 0.1 \\ -1.1 \\ 3.7 \\ 2.5 \end{bmatrix} \qquad c(t_1) = \begin{bmatrix} 1.5 \\ 2.9 \\ -0.3 \\ 2.0 \end{bmatrix}$$

Continuous-time HiPPO ODE

$$\frac{d}{dt}c(t) = A(t)c(t) + B(t)u(t)$$

discretize

(4)

Discrete-time HiPPO Recurrence

$$c_k = \overline{\boldsymbol{A}}c_{k-1} + \overline{\boldsymbol{B}}u_k$$

State Space Models (SSM)

Inconvénient de hippo : on passe d'un signal $u(t)$ 1-D à un état $c(t)$ N-D :

Solution : SSM extraire un signal $y(t)$ 1-D de l'état avec une projection affine de l'état $c(t)$:

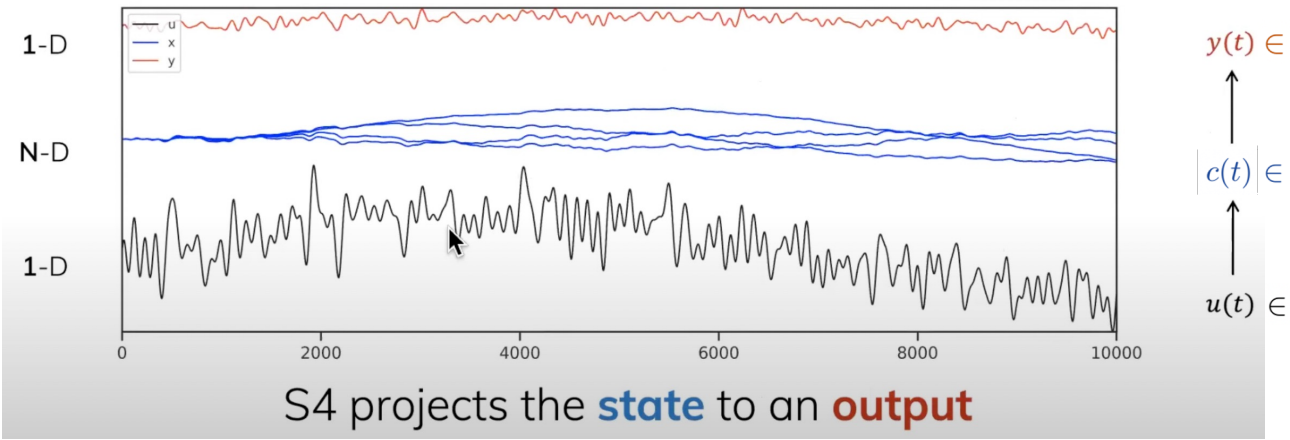
SSM cas continu (ED)

$$\begin{aligned} c'(t) &= \boldsymbol{A}c(t) + \boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}c(t) + \boldsymbol{D}u(t) \end{aligned}$$

Récurrence cas discret

$$\begin{aligned} c_k &= \overline{\boldsymbol{A}}c_{k-1} + \overline{\boldsymbol{B}}u_k \\ y_k &= \overline{\boldsymbol{C}}c_k + \overline{\boldsymbol{D}}u_k \end{aligned}$$

$$y(t) = \boldsymbol{C}c(t) + \boldsymbol{D}u(t)$$



Motivation : (Théorème) pour toute mesure $\mu^{(t)}$ raisonnable, l'état $c(t)$ vérifie une équation $c'(t) = \boldsymbol{A}c(t) + \boldsymbol{B}u(t)$

Idée : apprendre implicitement la mesure $\mu^{(t)}$ via $\boldsymbol{A}$ et $\boldsymbol{B}$ plutôt que la prescrire.

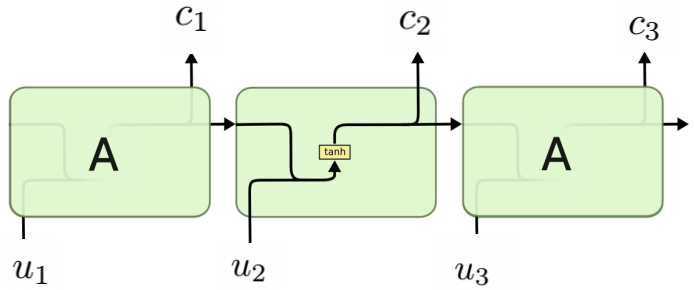
Difficulté : en pratique il est computationnellement impossible d'apprendre $\boldsymbol{A}$, en tous cas naïvement.

SSM : les vues récurrentes et convolutionnelles

Représentation récurrente

SSM $$\begin{aligned} c_k &= \overline{A}c_{k-1} + \overline{B}u_k \\ y_k &= \overline{C}c_k + \overline{D}u_k \end{aligned}$$

RNN



Mapping seq2seq **linéaire** $(u_k)_{l\in\mathbb{N}} \to (c_k)_{l\in\mathbb{N}}$

Mapping seq2seq **non-linéaire** $(u_k)_{l\in\mathbb{N}} \to (c_k)_{l\in\mathbb{N}}$

Efficace en prédiction, coût constant mais **non parallélisable**

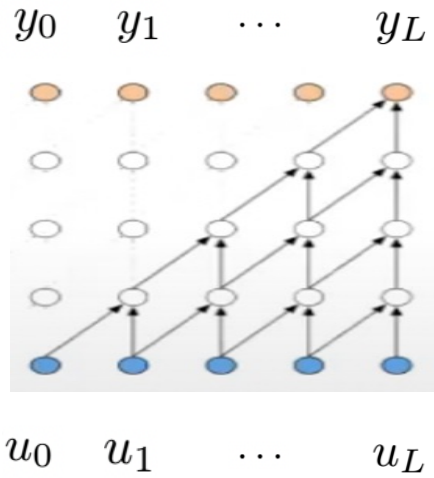
Représentation convolutionnelle (on déroule le SSM)

$$c_0 = \overline{B}u_0 \qquad c_1 = \overline{AB}u_0 + \overline{B}u_1 \qquad c_2 = \overline{A}^2\overline{B}u_0 + \overline{AB}u_1 + \overline{B}u_2 \qquad \dots$$

$$y_0 = \overline{CB}u_0 \qquad y_1 = \overline{CAB}u_0 + \overline{CB}u_1 \qquad y_2 = \overline{CA}^2\overline{B}u_0 + \overline{CAB}u_1 + \overline{CB}u_2 \qquad \dots$$

$$y_k = \overline{CA}^k\overline{B}u_0 + \overline{CA}^{k-1}\overline{B}u_1 + \cdots + \overline{CAB}u_{k-1} + \overline{CB}u_k$$

$$y = \overline{K} * u \qquad \overline{K} := (\overline{CB}, \overline{CAB}, \dots, \overline{CA}^{L-1}\overline{B}) \in \mathbb{R}^L$$ un très gros filtre CNN



Efficace à l'entraînement (futur connu) mais avec **un coût qui croit avec la taille du contexte**

SSM pour $\boldsymbol{A}$ quelconque est inapprenable en pratique!

La précision des prédictions du modèle SSM dépend énormément de $\boldsymbol{A}$

Un SSM avec une initialisation aléatoire de $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ est très **peu performant :**
Sur un benchmark classique (*sequential MNIST*) : précision misérable de **60%**

Mais avec $\boldsymbol{A}$ initialisé avec la **matrice HiPPO** : précision SOTA de **98%** !

Le coût d'entraînement d'un modèle SSM est prohibitif si l'on procède naïvement

Chaque multiplication par $\boldsymbol{A}$ coute $O(N^2)$. Pour une séquence de longueur L on a :

- Un coût de calcul $O(LN^2)$ pour calculer le filtre $\overline{\boldsymbol{K}} := (\overline{\boldsymbol{CB}}, \overline{\boldsymbol{CAB}}, \dots, \overline{\boldsymbol{CA}}^{L-1}\overline{\boldsymbol{B}},) \in \mathbb{R}^L$
- Une empreinte mémoire de $O(LN)$ pour stocker $(c_k)_{k\in[L]}$.

S4 = SSM + HiPPO + algo pour matrices $\boldsymbol{A}$ de type NPLR

S4 = Structured State Space Sequence Model

Théorème 1 : toutes les matrices HIPPO issues des mesures classiques $\mu^{(t)}$ sont **NPLR** (**N**ormal **P**lus **L**ow **R**ank).

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\dagger} + \boldsymbol{P}\boldsymbol{Q}^{\top}, \quad \boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_N] \in \mathbb{C}^N$$

$$\boldsymbol{U} \in \mathbb{C}^{N \times N}, \quad \boldsymbol{U}\boldsymbol{U}^{\dagger} = \boldsymbol{U}^{\dagger}\boldsymbol{U} = \mathbb{I}$$

$$\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{R}^{N \times r}, \quad r = 1, 2$$

Théorème 2 (S4 récurrent) : un pas de récurrence coûte $O(N)$. …au lieu de $O(N^2)$ naïvement !

Théorème 3 (S4 convolution) : le filtre $\overline{\boldsymbol{K}}$ demande $\tilde{O}(N + L)$ opérations et un espace mémoire de $O(N + L)$.

S4 : intégration dans une architecture de RN

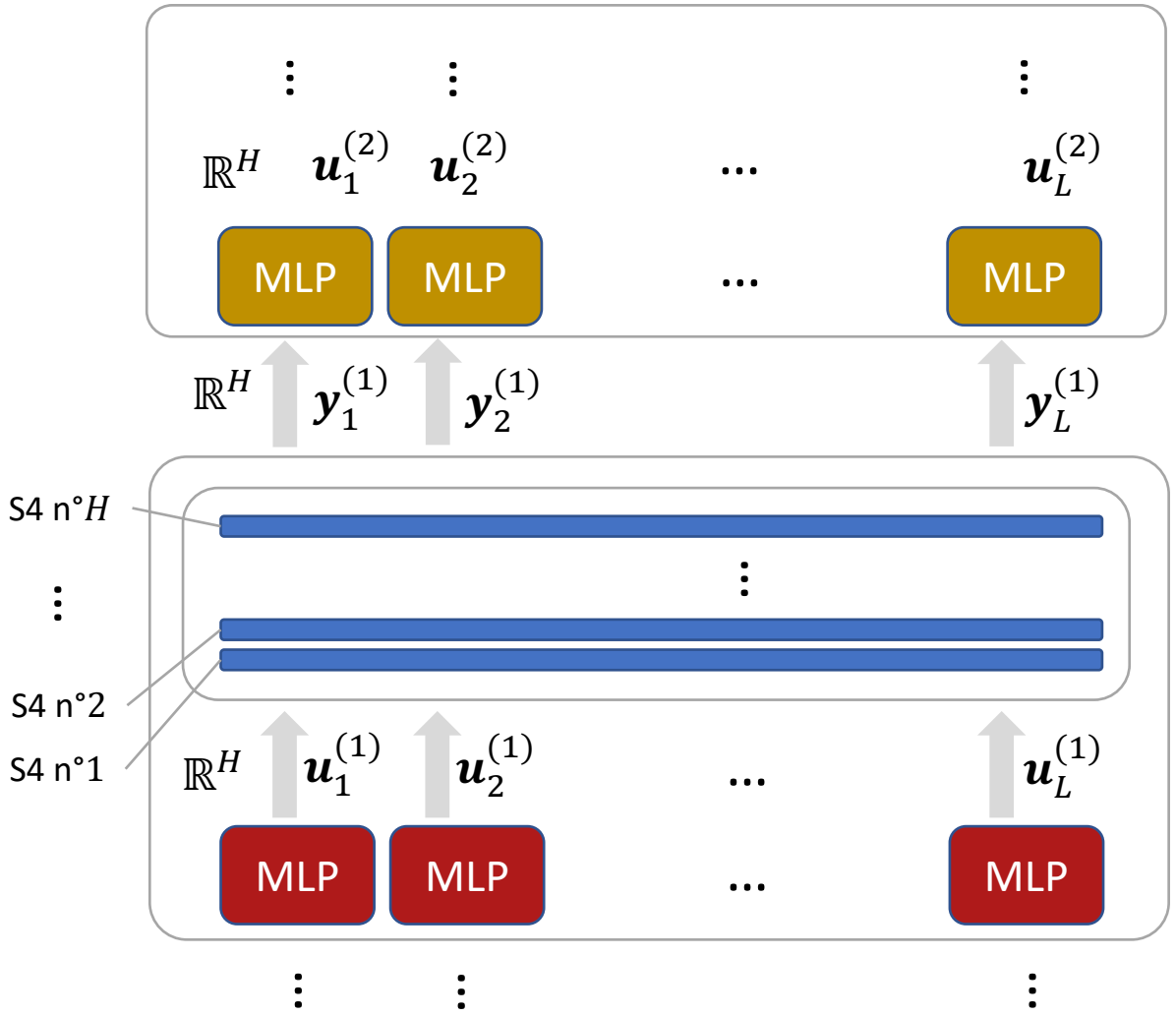
Les paramètres de S4 :

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\dagger} + \boldsymbol{P}\boldsymbol{Q}^{\top} = \boldsymbol{U}[\boldsymbol{\Lambda} - \underbrace{(\boldsymbol{U}^{\dagger}\boldsymbol{P})}_{\widetilde{\boldsymbol{P}}}\underbrace{(\boldsymbol{U}^{\dagger}\boldsymbol{Q})^{\dagger}}_{\widetilde{\boldsymbol{Q}}^{\dagger}}]\boldsymbol{U}^{\dagger}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_N] \in \mathbb{C}^N$$
$$\boldsymbol{B} \in \mathbb{C}^N, \quad \boldsymbol{C} \in \mathbb{C}^N, \quad \widetilde{\boldsymbol{P}} \in \mathbb{C}^N, \quad \widetilde{\boldsymbol{Q}} \in \mathbb{C}^N.$$

} $5N$ params

Si on veut un mapping entre des séquences de vecteurs $\in \mathbb{R}^H$ plutôt que des séquences de nombres on instancie simplement H modèles S4 indépendants !



S4 : à retenir

1. Un modèle de séquence générique (= ne demande aucun biais inductif, aucun ajustement d'architecture ou aucun pré-traitement) applicable à tous types des séquences (textes, images, audio, séries temporelles, ADN,…)
2. S4 : SSM + HiPPO + algo optimisé tirant parti de la structure NPLR des matrices $\boldsymbol{A}$ « utiles »
   Les matrices « utiles » sont précisément celles pour lesquelles le calcul est faisable !
3. Une approche du problème des LRD solidement motivée théoriquement : « *No dirty tricks!* »
4. Repose sur la paramétrisation judicieuse NPLR des matrices $\boldsymbol{A}$ d'un SSM suggérées par HiPPO
5. Très performant en pratique grâce à :
   - un entraînement efficace avec la vue CNN parallélisable $O(L + N)$
   - une prédiction rapide avec la vue RNN indépendante de la longueur L de la séquence
   - la robustesse vis-à-vis des fréquences d'échantillonnages (invariance d'échelle)
6. Une alternative rafraichissante aux Transformers !

Questions et discussion : S4 pour le TAL ?