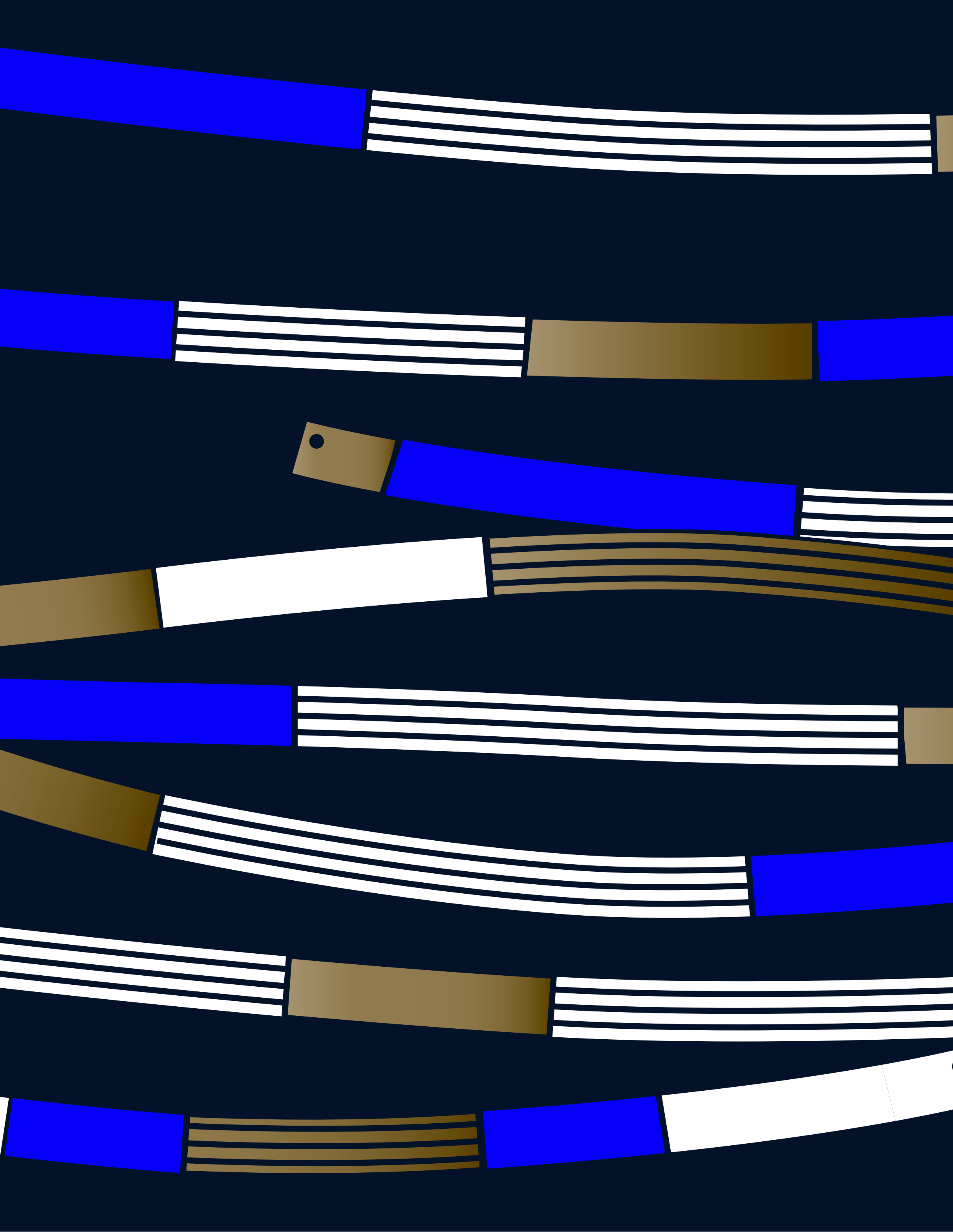
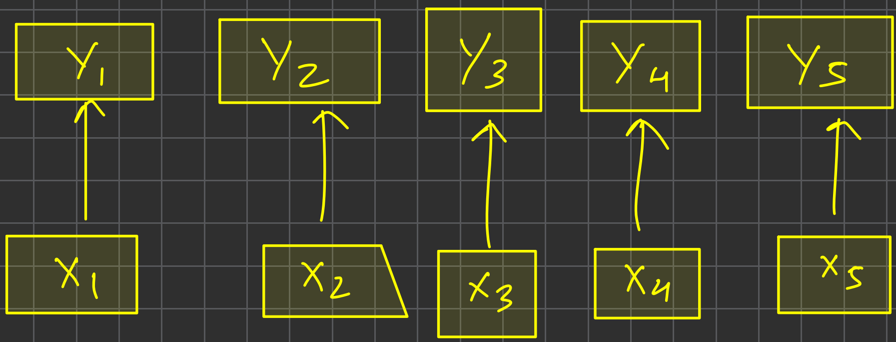


Sequence Modeling



Discrete Signal

Continuous Signal

What is Mamba ??

↓

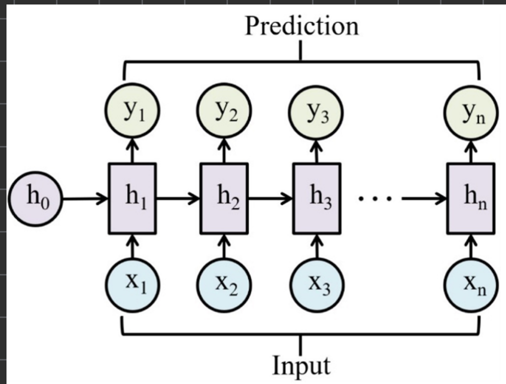
State Space model (SSM) + linear MLP (gated) + some selection mechanism

① RNNs → $O(LD)$

Simple RNN

L → context length (Sequence length)

D → model parameter or hidden size (d_{model})



① Memory Efficient

(Space Complexity)

① vanishing & Exploding gradients

② Limited context Size (can't retain Info)

③ Cannot be parallelized (Making training slow)

② CNN → $O(L^2D)$

- → Kernels
- → Convolution Filters

(Ex: UNET)

① context window is small

② Requires Stacking many Conv layers

Increasing computation cost

③ Transformers

① quadratic complexity → $O(L^2 D)$

② Easily parallelizable

③ Inference is slow (unless KV cache is used)

④ Limited vocab size / sequence length

* Differential Equation

$$\frac{db}{dt} = \lambda b(t)$$

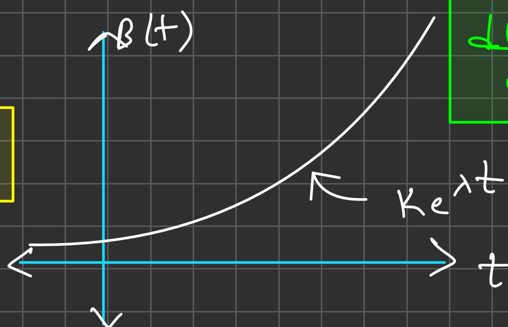
let $b(t) = Ke^{\lambda t}$

$$\frac{d(b(t))}{dt} = Ke^{\lambda t}(\lambda) = \lambda Ke^{\lambda t}$$

$$= \lambda Ke^{\lambda t}$$

$$\boxed{\frac{dB(t)}{dt} = \lambda B(t)}$$

$$\boxed{\dot{b} = \lambda b}$$



* State Space Model (linear & Time-Invariant)

objective : Map $x(t)$ to an output signal $y(t)$

$$h'(t) = A \cdot h(t) + B \cdot x(t)$$

$$y(t) = C \cdot h(t) + D \cdot x(t)$$

parameter matrices → A, B, C, D

(do not depend on Time)

$h(t)$ → state representation

$y(t)$ → output signal

→ Discretization

$h(0), h(1), h(2), h(3)$, etc

So finding $h(t)$ we want $h(t_k) = h(k\Delta)$

Δ → step size

$$\therefore b'(t) = \lambda b(t)$$

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

Δ → very small

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

$$b(t+\Delta) \cong b'(t)\Delta + b(t)$$

$$\therefore \underbrace{\boxed{b(t+\Delta) \cong \lambda b(t)\Delta + b(t)}}_{\text{recurrent Formulation}}$$

let $\lambda = 2$, $\Delta = 1$

① at $t=0$ $b=5$

$$b(0+1) \cong 2(b(0)1 + b(0)$$

$$\cong 2 \times 5 \times 1 + 5$$

$$b(1) \cong 15$$

$\therefore$ at $t=1$, $b=15$

$\therefore$ Discretization of our State State Model

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

$$\because h'(t) = A.h(t) + B.x(t)$$

$$h(t+\Delta) \cong (A.h(t) + B.x(t))\Delta + h(t)$$

$$\cong \Delta A h(t) + \Delta B.x(t) + h(t)$$

$$= h(t)(I + \Delta A) + \Delta B x(t)$$

$$\boxed{h(t+\Delta) = \bar{A} h(t) + \bar{B}.x(t)}$$

$$h(t+\Delta) \cong \bar{A}h(t) + \bar{B}.x(t)$$

$$\bar{A} = (I + \Delta A) \qquad \Delta \rightarrow \text{Time step}$$

$$\bar{B} = \Delta B \qquad I \rightarrow \text{Identity matrix}$$

In paper :-

upon solving

$$h'(t) = Ah(t) + B.x(t) \qquad h_t = \bar{A}h_{t-1} + \bar{B}x_t$$

$$y(t) = Ch(t) \qquad y_t = Ch_t$$

using zero-order hold (ZOH)

$$\bar{A} = f_A(\Delta, A) = e^{(\Delta A)}$$

$$\bar{B} = f_B(\Delta, A, B) = \frac{(e^{(\Delta A)} - I).\Delta B}{\Delta A}$$

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

$$y_t = Ch_t$$

parameters $\rightarrow \Delta, A, B, C$

(learnable)

Recurrent Computation

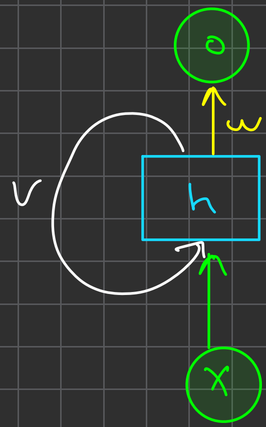
Initial condition → at $t=0$, $h_{t-1}=0$
(Initial state is zero)

$\therefore h_t = \bar{A}h_{t-1} + \bar{B}x_t \qquad y_t = Ch_t$

$\therefore t=0 \quad h_{t-1}=0$

$h_0 = 0 + \bar{B}x_0$

$y_0 = Ch_0$



$\therefore t=1$

$h_1 = \bar{A}h_0 + \bar{B}x_1$

$y_1 = Ch_1$

$\therefore t=2$

$h_2 = \bar{A}h_1 + \bar{B}x_2$

$y_2 = Ch_2$

RC → Memory efficient (during Inference)
↳ not good during Training

paralelizable ?

Convolutional Computation

$h_t = \bar{A} h_{t-1} + \bar{B} x_t$ at $\underbrace{t=0, h=0}_{\text{given}}$

$y_t = C h_t$

① $t=0$

$\therefore h_0 = \bar{A} \cancel{h}^{0} + \bar{B} x_0$

$h_0 = \bar{B} x_0$

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

② $t=1$

$h_1 = \bar{A} h_0 + \bar{B} x_1$

$h_1 = \bar{A}\bar{B} x_0 + \bar{B} x_1$

$y_1 = C h_1$

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

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

$\therefore y_K = \left[C(\bar{A})^K \bar{B} x_0 + C(\bar{A})^{K-1} \bar{B} x_1 + \dots + C\bar{A}\bar{B} x_{K-1} + C\bar{B} x_K \right]$

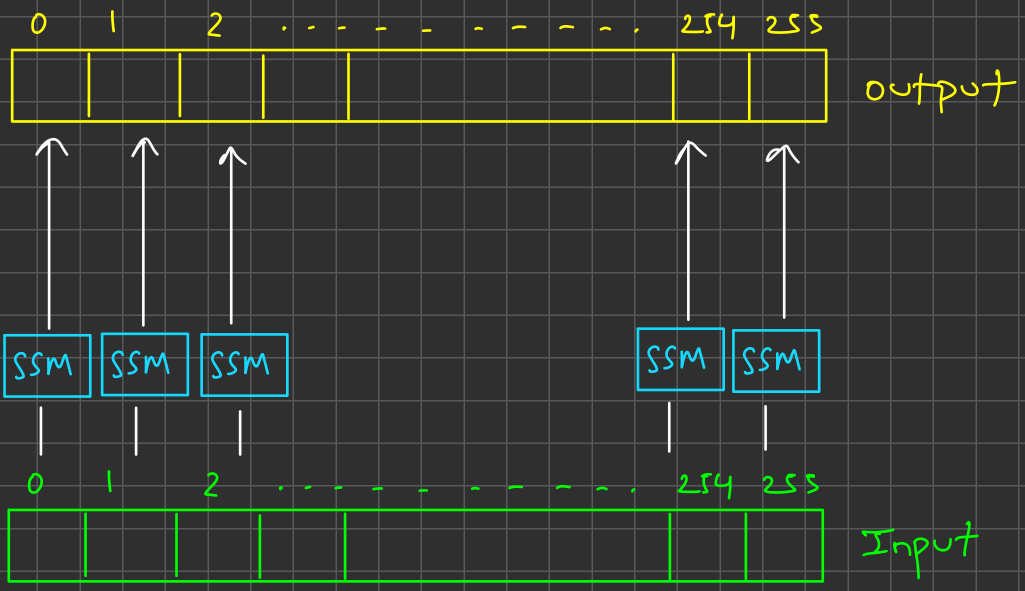
$\bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \dots, C(\bar{A})^K \bar{B}, \dots\dots)$ [3(a)]

Kernel

$y = x * \bar{K}$

- Can be parallelized : because output y_K doesn't depend on y_{K-1}, but only on the kernel & input

Multi-dimension SSM

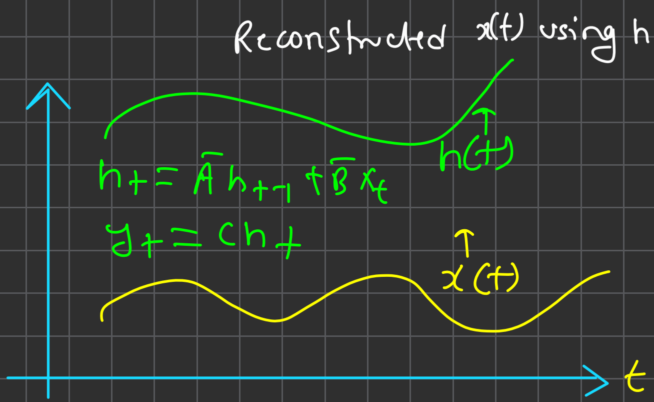


In this way we can parallelize all these operations by working on batches of input

A, B, C, D, x(t), y(t) become Tensors

HIPPO THEORY

With hippo theory, we Build the A matrix in such a way that it approximates all the Input Signal seen so far into a vector of coefficient (representing Legendre polynomials).



(Hippo Matrix)

↳ $(n \times k)$

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

$O(N \log N)$ order

Hippo LegS (Scaled Legendre)

- H3 (High-order polynomial projection operator Theory) is a mathematical framework designed to efficiently store & retrieve sequential Information by projecting data onto Special basis functions

For Example $N = 3$ (State Space Model size)

$A_{11} = n+1 = 3+1 = 4$ $(k = n)$

$A_{12} = 0$ $(k > n)$

$A_{13} = 0$ $(k > n)$

$A_{21} = (2n+1)^{1/2} (2k+1)^{1/2} = (2(2)+1)^{1/2} (2(1)+1)^{1/2}$

$\because (n > k)$ $= \sqrt{5}\sqrt{3} = \sqrt{15}$

$$A_{nk} = \begin{bmatrix} 4 & 0 & 0 \\ \sqrt{15} & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ \end{bmatrix}^T$$

Improving SSMs with Selection

Selection Mechanism

$SSM(\bar{A}, \bar{B}, C)(x)$

$$\boxed{h_t = \bar{A}(G \odot h_{t-1}) + \bar{B}(F \odot x(t))}$$

$\odot \rightarrow$ element wise multiplication

$G \rightarrow$ Selection gate controlling how much memory to retain

$F \rightarrow$ Feature gate, determines $x(t)$ in interacts with memory

Algorithm (1) : Dynamic Selection

Input : $x : (B, L, D)$, output : $y : (B, L, D)$

$A \rightarrow (D, N)$ parameter (Hippo-H_3)

$B \rightarrow (D, N)$ parameter

$C \rightarrow (D, N)$ parameter

$\Delta \rightarrow (D)$ step size

Compute $\bar{A}, \bar{B} : (D, N) \leftarrow$ discretize(Δ, A, B) (ZOH)

$y \leftarrow SSM(\bar{A}, \bar{B}, C, F, G)(x)$

(Time invariant) (recurrent + conv) can be used

gated Mechanism

Algorithm ②

Input: $x : (B, L, D)$, output : $y : (B, L, D)$

$A \in \mathbb{R}^{N \times N}$ $A \longrightarrow (D, N)$ parameter (HIPPO M_3)

$(B, L, N) \leftarrow (D, N)$

$B \in \mathbb{R}^{N \times D}$ $B \longrightarrow (B, L, N)$ $(B, L, N) \leftarrow S_B(x)$

trans forms the input Sequence into a 3D Tensor

$C \in \mathbb{R}^{B \times L \times N}$ $C \longrightarrow (B, L, N)$

$\Delta \longrightarrow (B, L, D)$ ← step size

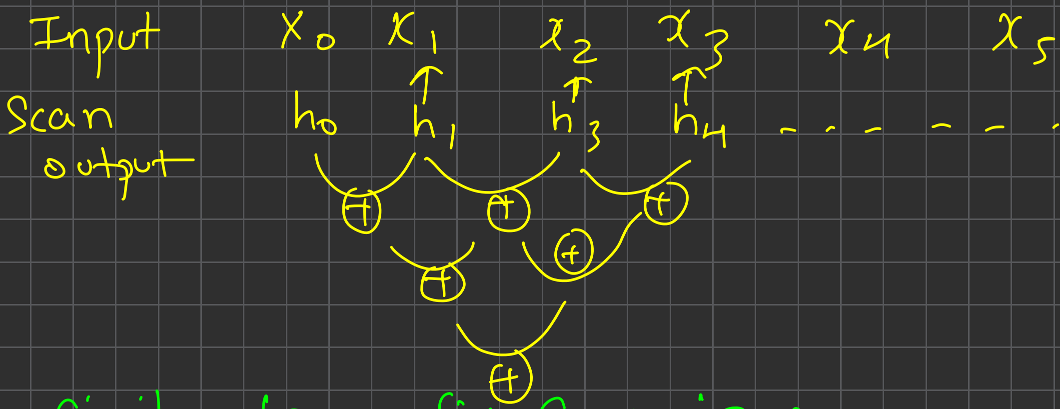
Compute $\bar{A}, \bar{B} : (B, L, D, N)$

↳ discretize(Δ, A, B)

$y \leftarrow SSM(\bar{A}, \bar{B}, C, F, \sigma)(x)$

↳ Time varying ∴ Recurrent (scan) only

The Scan operation



Similar to prefix sum in an array

Kernel Fusion :

When we perform a tensor operation, our deep Learning framework (eg pytorch) Loads the tensor in the fast memory (SRAM) of the GPU, performs the operations (e.g. matrix multiplication), & then saves back the result in the high Bandwidth Memory of the GPU

(use a single cuda kernel)