Phân tích mô hình Mamba

Hồ Nguyễn Phú

01/08/2025

Abstract:

Bài báo Mamba xuất bản năm 2024 với tiêu đề "Mamba: Linear-Time Sequence Modeling with Selective State Spaces" (Gu & Dao, 2024). Theo tác giả bài báo, Mamba có thể được coi là sự kết hợp của mạng neural hồi quy (RNN) và mạng neural tích chập (CNN), lấy cảm hứng từ mô hình không gian trạng thái state space model (SSM). Từ những tồn tại hiện hữu của kiến trúc Transformers của Vaswani et al. (2017), Gu và Dao (2024) đã đề xuất mô hình Mamba với các đặc điểm sau: giữ được khả năng mô hình hóa của Transformers nhưng với khả năng mở rộng tuyến tính (scale linearly) với độ dài chuỗi (ở Transformers, các phép tính toán tăng bình phương theo độ dài chuỗi). Để tìm hiểu về Mamba, ta sẽ ôn lại và tìm hiểu về (1) ODE cơ bản (2) cơ bản về lý thuyết điều khiển (control theory), (3) SSM và mô hình S4, (4) Mamba – Selective SSM và Selective Scanning, (5) cài đặt môi trường và (6) thực hành với mô hình Mamba.

A. Toán cần thiết:

I. ODE:

1. Giới thiệu:

- Các hiện tượng thực tế thường có thể được biểu diễn bằng đạo hàm, e.g., vận tốc, gia tốc. Do đó, người ta dùng phương trình vi phân thường, Ordinary Derivative Equation (ODE) cho mục đích mô hình hóa (Kreyszig et al., 2011, trang 2).
- **Định nghĩa**: Phương trình vi phân thường là phương trình chứa một hoặc nhiều đạo hàm của một hàm số chưa biết, thường gọi là y(x) (đôi khi là y(t) nếu y phụ thuộc vào biến thời gian t).
- Bên cạnh các đạo hàm của y, ODE có thể chứa chính y, các phương trình của x hoặc hằng số. Ví dụ:
  - $y' = cos(x)$
  - $y'' + 9y = e^{-2x}$
  - State Space (appendix [1]):

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) & \text{(phương trình trạng thái)} \\ y(t) = Cx(t) + Du(t) & \text{(phương trình đầu ra)} \end{cases}$$

- Mamba có SSM là một phương trình ODE bậc nhất, tức đạo hàm bậc cao nhất của y là bậc một (Kreyszig et al., 2011, trang 4):

$F(x, y, y') = 0$

- Phần về ODE liên quan đến Mamba trong sách của Kreyszig et al. (2011) bao gồm chương 1 (ODE bậc 1) và chương 6 (Laplace transform – cho xử lý ODE tuyến tính).

2. Chương 1 - ODE bậc một:

- Bàn về nghiệm của phương trình ODE (Kreyszig et al., 2011):
- Nghiệm của phương trình ODE trên khoảng $a < x < b$ là hàm số $y = h(x)$ sao cho $h(x)$ **có nghĩa** và **có nghiệm** trên khoảng $(a;b)$ và sao cho khi thế y và y' bằng h và h' ta có được hàm đồng nhất (identity function) với hàm ODE ban đầu.
- Biểu đồ của hàm h(x) là biểu đồ nghiệm (solution curve).
- Nghiệm chứa **hằng số *c*** của phương trình ODE được gọi là **general solution**, trong khi nghiệm với **c bằng một giá trị cụ thể** (e.g., $c = 23$) được gọi là **particular solution**.
- Xem thêm ví dụ về exponential growth và exponential decay (Kreyszig et al., 2011, trang 5-6).
- Bài toán giá trị ban đầu (Initial Value Problem):
- Bài toán: tìm particular solution với general solution $y' = f(x, y)$ bằng **điều kiện ban đầu (initial condition)**: $y(x_0) = y_0$
- Hàm số đặc biệt: $y = a.e^{cx}$ có đạo hàm $y' = a.c.e^{cx} = c.y \rightarrow$ liên quan đến Exponential kernel trong SSM.

a) Ý nghĩa hình học của ODE và phương pháp Euler:

- Ta có ODE bậc một: $y' = f(x, y)$
- Ta thấy: $y'(x)$ là độ dốc của $y(x)$, vậy nên $f(x_0, y_0)$ sẽ là độ dốc của y tại điểm (x_0, y_0).
- Direction field có thể vẽ thông qua các mũi tên chỉ slope của hàm số y'(x) tại các điểm.
- Phương pháp Euler là nền tảng cho việc rời rạc hóa công thức ODE (bàn thêm ở phần sau).
  - + Đặt $h = (x_{i+1} - x_i)$
  - + Với (x_0, y_0) ban đầu, ta xét:

$$\begin{aligned} y_{i+1} &= y_i + (y_{i+1} - y_i) \\ &= y_i + y_i' \\ &= y_i + f(x_i, y_i)(x_{i+1} - x_i) \\ &= y_i + h.f(x_i, y_i) \\ &= y_i + h.f(x_i, y(x_i)) \end{aligned}$$

b) Phương trình vi phân tách biến (Separable ODE):

- Nhiều hàm ODE hữu ích có thể đưa về dạng:

  $g(y)y' = f(x)$

- Sau đó ta nguyên hàm cả hai vế và có được:

  $\int g(y)y'dx = \int f(x)dx + c$

  $\Leftrightarrow \quad \int g(y)\frac{dy}{dx}dx = \int f(x)dx + c$

  $\Leftrightarrow \quad \int g(y)\,dy = \int f(x)dx + c$

- Phương trình trên được gọi là phương trình tách biến vì các biến y và x được tách ra ở hai vế phương trình.

c) ODE tuyến tính (SSM):

- ODE tuyến tính hoặc ODE có thể biến đổi thành dạng tuyến tính được dùng làm mô hình của nhiều hiện tượng.
- Một ODE bậc một được gọi là tuyến tính nếu nó có dạng:

  $y' + p(x)y = r(x)$

- Đặc trưng của ODE tuyến tính là:
  - + Cả hàm chưa biết y và đạo hàm y' đều tuyến tính
  - + p và r là bất kỳ hàm số nào của x
- Với hàm $y' + p(x)y = r(x)$ thì $r(x)$ thường được xem là input, còn $y(x)$ là output tương ứng với input và điều kiện ban đầu (*initial condition*).

● **Giải:** Phương trình vi phân thường thuần nhất (Homogeneous Linear ODE)

- Với $r(x) = 0$, ta có:

  $y' + p(x)y = 0$

  $\Leftrightarrow \quad y' = -p(x)y$

  $\Leftrightarrow \quad \frac{dy}{dx} = -p(x)y$

  $\Leftrightarrow \quad \frac{dy}{y.dx}dx = -p(x)dx$

  $\Leftrightarrow \quad \frac{dy}{y} = -p(x)dx$

  $\Leftrightarrow \quad \ln|y| = -\int p(x)dx + c^*.$

  $\Leftrightarrow \quad e^{\ln|y|} = e^{-\int p(x)dx + c^*}$

  $\Leftrightarrow \quad y = e^{c^*}.\, e^{-\int p(x)dx}$

  $\Leftrightarrow \quad y = c\,.\, e^{-\int p(x)dx}$

- Chọn $c = 0$, nghiệm tầm thường (trivial solution) của phương trình là y(x) = 0 với mọi x trong khoảng cho sẵn.

- **Giải:** Tìm nghiệm tổng quát phương trình vi phân thường không thuần nhất (Nonhomogeneous Linear ODE).

- Ta có:

$y' + py = r$

$\Leftrightarrow \quad Fy' + Fpy = Fr$

$\Leftrightarrow \quad Fy' + F'y = Fr \qquad (Với\ pF = F') \qquad (1)$

- *Đặt $h = ln|F|$, Ta có:*

$pF = F'$

$\Leftrightarrow \quad pF = \frac{dF}{dx}$

$\Leftrightarrow \quad p = \frac{dF}{Fdx}$

$\Leftrightarrow \quad pdx = \frac{dF}{F}$

$\Leftrightarrow \quad \int \frac{1}{F} dF = \int p\,dx + c$

$\Leftrightarrow \quad ln|F| = \int p\,dx + c$

$\Leftrightarrow \quad h = \int p\,dx + c$

$\Leftrightarrow \quad F = e^h \qquad (2)$

- *Từ (1) và (2) ta suy ra:*

$e^h y' + (e^h)' y = e^h r$

$\Leftrightarrow \quad (e^h y)' = e^h r$

$\Leftrightarrow \quad e^h y = \int e^h r\,dx + c$

$\Leftrightarrow \quad y = e^{-h}(\int e^h r\,dx + c)$

$\Leftrightarrow \quad y = (e^{-h} \int e^h r\,dx) + (e^{-h} c)$

- *Phân tích:*

Total Output = Response to Input r + Response to initial data

- Từ đây, ta sẽ phân tích phương trình SSM.

3. Xem thêm:

- Phần còn lại chương 1 (Kreyszig et al., 2011, trang 1-44).
- Chương 6 (Laplace transform).

II. Cơ bản về lý thuyết điều khiển:

1. Linear System:

- Một hệ thống là tuyến tính khi và chỉ khi thỏa mãn nguyên lý chập chồng (*superposition principle*).

$$F(x_1 + x_2) = F(x_1 + x_2)$$

$$F(ax) = aF(x)$$

- Phương trình:

-

Phương trình ODE tuyến tính	Phương trình Linear System
$y'(x) + p(x)y = r(x)$	$y'(t) = Ax(t) + f(t)$
$y(x) = (e^{-h(x)} \int e^{h(x)} r(x)dx) + (e^{-h(x)} c)$	$x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)} f(s)ds$

2. Tìm hiểu thêm sau:

III. Mô hình không gian trạng thái (State Space Model – SSM) và S4:

1. Tổng quan:

- "State Space Model là một cách tiếp cận để biểu diễn và giải quyết các vấn đề trong AI. Trong mô hình này, chúng ta biểu diễn trạng thái (state) của hệ thống và cách nó thay đổi theo thời gian. Hãy tưởng tượng bạn đang chơi cờ vua: mỗi trạng thái là một vị trí các quân cờ trên bàn cờ và mỗi lần di chuyển là một sự thay đổi trạng thái" (Nguồn: Facebook).
- Mô hình Structured State Space Sequence (S4) có liên hệ đến RNNs, CNNs, và các mô hình State Space cổ điển (Gu, 2022).
- Đặc điểm của các mô hình SSM là tạo ra một latent state – có thể hiểu như một không gian đặc trưng lớn hơn so với không gian của embedding đầu vào (Gu & Dao, 2024). Theo cách hiểu của người viết (không trích dẫn) thì không gian này như một nơi lưu trữ thông tin trạng thái của các bước trước đó, và liên tục được cập nhật bởi đầu vào qua các bước.
- Các yếu tố ta quan tâm trước khi tìm hiểu mô hình Mamba:
  - + Trực giác (*intuition*) đằng sau công thức SSM
  - + Rời rạc hóa phương trình liên tục
  - + Tích chập cho SSM

2. **Phân tích công thức:**

- Với nền tảng phương trình ODE tuyến tính và phương pháp Euler, ta có được phương trình State-Space transition (Wiki) – tạm bỏ qua vector *w(t)* miêu tả *additive disturbances*.

- **Phân tích phương trình ODE tuyến tính so với SSM:**

- Với phương trình ODE tuyến tính không thuần nhất:

$$y' + p(x)y = r(x)$$
$$\Leftrightarrow \quad y' = -p(x)y + r(x) \qquad (3)$$

- **Đặt:**

Phương trình ODE tuyến tính	**Phương trình State Space**
$y' = -p(x)y + r(x)$	$x'(t) = Ax(t) + Bu(t)$
$y(x)$	$x(t)$
$-p(x)$	A
$r(x)$	$Bu(t)$

- Từ (3) ta suy ra phương trình State Space của mô hình S4 (?):

$$x'(t) = Ax(t) + Bu(t)$$

- Ta thấy, khác với phương trình ODE tuyến tính, ma trận A của State Space (và Linear System nói chung) trong trường hợp trên không phụ thuộc vào đầu vào thời gian *t (time-invariant)*, do đó State Space model là hệ thống tuyến tính bất biến theo thời gian (LTI).

- **Phân tích nghiệm của phương trình ODE tuyến tính so với SSM:**

$$y(x) = (e^{-h(x)} \int e^{h(x)} r(x) dx) + (e^{-h(x)} c)$$

- Với $h(x) = \int p(x)\, dx + c$

- **Đặt:**

Nghiệm phương trình ODE tuyến tính	Nghiệm phương trình State Space
$y(x) = (e^{-h(x)} \int e^{h(x)} r(x)dx) + (e^{-h(x)} c)$	$y = Cx(t) + Du(t)$
$e^{-h(x)} \int e^{h(x)} r(x)dx$	$Cx(t)$
$e^{-h(x)} c$	$Du(t)$

- **Như vậy:**
- Nền tảng lý thuyết của phương trình State space là phương trình ODE tuyến tính.
- Chuyển từ phương trình SSM liên tục thành phương trình SSM rời rạc (AI Việt Nam, 2024):

$$
\begin{aligned}
h_{t+\Delta} &\approx (\mathbf{A}h_t + \mathbf{B}x_t)\Delta + h_t \\
h_{t+\Delta} &\approx \Delta\mathbf{A}h_t + \Delta\mathbf{B}x_t + h_t \\
h_{t+\Delta} &\approx (\Delta\mathbf{A} + \mathbf{I})h_t + \Delta\mathbf{B}x_t \\
h_{t+\Delta} &\approx \overline{\mathbf{A}}h_t + \overline{\mathbf{B}}x_t,
\end{aligned}
$$

3. **Phân tích mô hình S4**: Rời rạc hóa phương trình liên tục và tích chập
- Xem thêm: Wikipedia.
- **Rời rạc hóa (discretization)**
- Phương trình SSM là phương trình liên tục (Continuous-time SSM) với dữ liệu liên tục, như đầu vào *u(t)*. Trong khi đó, máy tính xử lý dữ liệu rời rạc.
- Giả định đầu vào thực tế u_k được lấy mẫu (*sampling*) từ một tín hiệu liên tục *u(t)*, với $u_k = u(k.\Delta)$. Trong đó, Δ là bước thời gian (tức $k.\Delta$ trở thành thời gian tương ứng với chỉ mục / index *k*).
- Tác giả S4 (Gu, 2022) sử dụng bilinear method nhằm chuyển ma trận trạng thái A thành xấp xỉ $\overline{A}$ (ma trận D tạm đặt bằng 0).

$$
\begin{aligned}
x_k &= \overline{\boldsymbol{A}}x_{k-1} + \overline{\boldsymbol{B}}u_k \quad & \overline{\boldsymbol{A}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}(\boldsymbol{I} + \Delta/2 \cdot \boldsymbol{A}) \\
y_k &= \overline{\boldsymbol{C}}x_k & \overline{\boldsymbol{B}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}\Delta\boldsymbol{B} \qquad & \overline{\boldsymbol{C}} &= \boldsymbol{C}.
\end{aligned}
\tag{4}
$$

- Như vậy, phương trình trên ánh xạ *"sequence-to-sequence"* thay vì *"function-to-function"* như phương trình SSM gốc.
- x_k có thể được xem như hidden state với ma trận chuyển tiếp $\overline{A}$.

- **Huấn luyện SSM**: biểu diễn dưới dạng tích chập

- Do đặc điểm tuần tự của SSM hồi quy (*recurrent SSM*), nó không thích hợp với phần cứng hiện đại (khả năng song song hóa khi huấn luyện – một điểm yếu của các mô hình tuần tự như RNN). Ngược lại, các mạng SSM bất biến với thời gian (*LTI SSM*) có liên hệ với tích chập liên tục (*continuous convolution)*. Từ đây, phương trình (4) có thể biểu diễn dưới dạng tích chập rời rạc.
- Để dễ hình dung, ta thực hiện phương trình (4), hay chi tiết hơn là phương trình $x_k = \overline{A}x_{k-1} + \overline{B}u_k$ qua các bước thời gian như sau:

Trạng thái x_{k-1}	**Trạng thái** $x_k = \overline{A}x_{k-1} + \overline{B}u_k$
$x_0 = 0$	$x_1 = 0\overline{A} + \overline{B}u_0$
$x_1 = \overline{B}u_0$	$x_2 = \overline{AB}u_0 + \overline{B}u_1$
$x_2 = \overline{AB}u_0 + \overline{B}u_1$	$x_3 = \overline{AAB}u_0 + \overline{AB}u_1 + \overline{B}u_2$
$x_3 = \overline{AAB}u_0 + \overline{AB}u_1 + \overline{B}u_2$	$x_4 = \overline{AAAB}u_0 + \overline{AAB}u_1 + \overline{AB}u_2 + \overline{B}u_3$
$x_4 = \overline{AAAB}u_0 + \overline{AAB}u_1 + \overline{AB}u_2 + \overline{B}u_3$	…

- Tương tự, áp dụng cho đầu ra $y_k = \overline{C}x_k$, ta có phép tích chập $y = K * u$

$$\overline{K} = (\overline{CB},\ \overline{CAB},\ \overline{CAAB},\ \overline{CAAAB},\ \ldots)$$

- $\overline{K}$ ở đây được gọi là SSM convolution kernel và là đóng góp chính (về mặt tối ưu tính toán) của bài báo S4 (Gu, 2022). Sơ lược:

Algorithm 1 S4 Convolution Kernel (Sketch)

Input: S4 parameters $\boldsymbol{\Lambda}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^N$ and step size Δ
Output: SSM convolution kernel $\overline{\boldsymbol{K}} = \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ for $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{Q}^*$ (equation (5))

1: $\widetilde{\boldsymbol{C}} \leftarrow \left(\boldsymbol{I} - \overline{\boldsymbol{A}}^L\right)^* \overline{\boldsymbol{C}}$ ▷ Truncate SSM generating function (SSMGF) to length L

2: $\begin{bmatrix} k_{00}(\omega) & k_{01}(\omega) \\ k_{10}(\omega) & k_{11}(\omega) \end{bmatrix} \leftarrow \left[\widetilde{\boldsymbol{C}}\ \boldsymbol{Q}\right]^* \left(\frac{2}{\Delta}\frac{1-\omega}{1+\omega} - \boldsymbol{\Lambda}\right)^{-1} [\boldsymbol{B}\ \boldsymbol{P}]$ ▷ Black-box Cauchy kernel

3: $\hat{\boldsymbol{K}}(\omega) \leftarrow \frac{2}{1+\omega}\left[k_{00}(\omega) - k_{01}(\omega)(1 + k_{11}(\omega))^{-1}k_{10}(\omega)\right]$ ▷ Woodbury Identity

4: $\hat{\boldsymbol{K}} = \{\hat{\boldsymbol{K}}(\omega) : \omega = \exp(2\pi i \frac{k}{L})\}$ ▷ Evaluate SSMGF at all roots of unity $\omega \in \Omega_L$

5: $\overline{\boldsymbol{K}} \leftarrow \mathsf{iFFT}(\hat{\boldsymbol{K}})$ ▷ Inverse Fourier Transform

IV. Mamba – Selective SSM và Selective Scanning:

1. Selective SSM:

a) Bối cảnh:

+ Ta có thể thấy, S4 giúp tăng hiệu quả tính toán thông qua tích chập,
+ Tuy nhiên, các ma trận A, B, C, D ở đây có đặc điểm bất biến với thời gian (time-invariance). Nguyên nhân: hạn chế về tính toán.
+ Gu và Dao (2024) đã giới thiệu (lại) cơ chế phụ thuộc vào input (input-dependent) cho các ma trận này trong mô hình Mamba.
+ Điều này yêu cầu cải tiến về mặt thuật toán để thích hợp với phần cứng.

b) Thông tin chung về thuật toán:

- Rời rạc hóa đầu vào:
  + Sử dụng zero-order hold (ZOH).
  + Phương trình:

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

  + Các ma trận rời rạc sẽ là ma trận chéo (diagonal) do: "… structured SSMs are so named because computing them efficiently also requires imposing structure on the $\boldsymbol{A}$ matrix. The most popular form of structure is diagonal…"
- Khi có được ma trận rời rạc rồi, ta có thể tính toán theo hai cách: *(a) linear recurrence* hoặc *(b) global convolution*. Phương pháp thường dùng là *(a)* cho huấn luyện và *(b)* cho inference (autoregressive inference).
- Flow chart Selective SSM:

Algorithm 1 SSM (S4)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
$\boldsymbol{A} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
$\boldsymbol{C} : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
$\Delta : (\mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter})$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
return y

Algorithm 2 SSM + Selection (S6)

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

+ $s_B(x) = Linear_N(x)$
+ $s_C(x) = Linear_N(x)$
+ $s_\Delta(x) = Broadcast_D(Linear_1(x))$
+ $\tau_\Delta = softplus = ln(1 + e^x)$

c) Phân tích Selective SSM:

- **Ta có các chiều**: B (batch), L (length), D (input embedding dimension) và N (latent state dimension).
- **Ý nghĩa từng ma trận:**

- Ma trận A: ma trận chuyển đổi giúp cập nhật state (nhân với state)
- Ma trận B: ma trận ánh xạ input để cập nhật state
- Ma trận C: ma trận ánh xạ state thành output
- Ma trận D: một dạng skip connection – bổ sung identity của input vào phép tính đầu ra y
- Ma trận Δ: tính toán từ bước thời gian (*time step*)
    + Về mặt ý tưởng, Δ ảnh hưởng quá trình cập nhật trạng thái.
    + Bước thời gian Δ xuất phát từ Lý thuyết điều khiển (?). Trong đó, bước thời gian càng nhỏ thì mô hình càng gần với "liên tục" và ngược lại.
    + Với Δ nhỏ, mô hình sẽ "giữ lại" nhiều thông tin từ trạng thái trước đó. Còn với Δ lớn, mô hình sẽ "quên" trạng thái trước đó nhanh hơn – nói cách khác, tăng trọng số của thông tin input mới.
    + Theo tác giả, Selection Mechanism là dạng tổng quát hơn của cơ chế "cổng kiểm soát" RNN cổ điển (*classical gating mechanism*). Cơ chế này sử dụng *sigmoid* để ánh xạ input thành khoảng *[0, 1]*, từ đó quyết định thông tin mới này được giữ lại bao nhiêu trong tương quan với trạng thái.

$$g_t = \sigma(\text{Linear}(x_t))$$
$$h_t = (1 - g_t)h_{t-1} + g_t x_t.$$

    + Với Selective SSM thì ma trận bước thời gian Δ được xử lý như sau:
    (1) Biến đổi tuyến tính (đầu ra 1 chiều)
    (2) Broadcast lên N chiều tương ứng với N chiều của latent state (đảm bảo ảnh hưởng lên tất cả các phép tính)
    (3) Cộng với bias (Parameter) – đảm bảo ổn định khi huấn luyện (?)
    (4) Đưa vào hàm Softplus – một dạng của ReLU, trong đó các giá trị luôn dương và trả về giá trị tăng tuyến tính khi đầu vào tiến đến dương vô cực (mô phỏng bước thời gian luôn dương).

$$\Delta_t = \tau_\Delta(\text{Parameter} + s_\Delta(x_t))$$

    + Sau đó, ma trận Δ được sử dụng để tính toán ma trận $\overline{A}$ và $\overline{B}$ – tương đương g_t và $1 - g_t$ trong công thức *gating mechanism* của RNN.

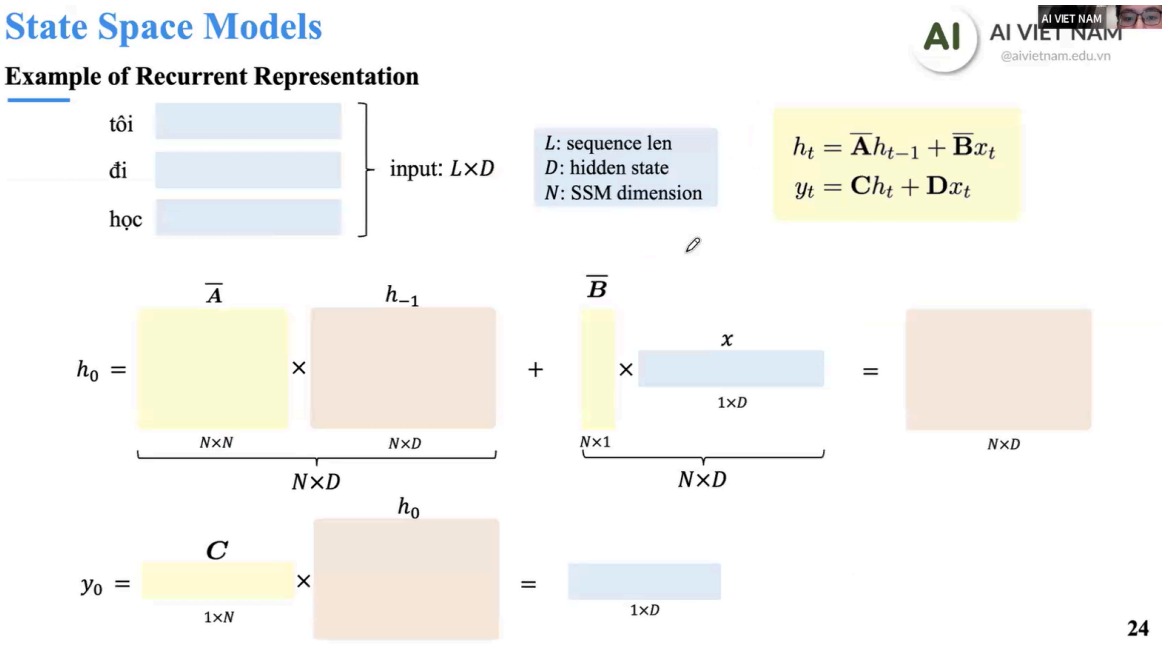
- **Phân tích các bước**: Ta phân chia các bước cho dễ hiểu hơn như sau.

- **Bước 1**: Tạo nên các ma trận phụ thuộc đầu vào B, C và Δ
  + Ma trận A ban đầu là cố định.
  + Ma trận B và C sẽ được quyết định thông qua lớp tuyến tính xử lý input (đầu ra lớp này có N chiều – tương ứng N số trong ma trận chéo B và C).
  + Ma trận Δ được tính toán như trên.
- **Bước 2:** Tính toán ma trận rời rạc $\overline{A}$ và $\overline{B}$

$$\overline{\boldsymbol{A}}_t = \exp(\Delta \boldsymbol{A}) = \frac{1}{1+\exp(\text{Linear}(x_t))} = \sigma(-\text{Linear}(x_t))$$
$$= 1 - \sigma(\text{Linear}(x_t))$$
$$\overline{\boldsymbol{B}}_t = (\Delta \boldsymbol{A})^{-1}(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I}) \cdot \Delta \boldsymbol{B} = -(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I}) = 1 - \overline{\boldsymbol{A}}$$
$$= \sigma(\text{Linear}(x_t)).$$

  + Tính thông qua áp dụng công thức rời rạc hóa zero-order hold (ZOH).
  + Cho ra kết quả tương đương với *gating mechanism* của RNN.

- **Về chiều dữ liệu (AI Vietnam, 2024):**



2. Selective Scanning:

a) Bối cảnh:

- Để bổ sung tính phụ thuộc vào thời gian vào mô hình, Mamba không còn có thể sử dụng cơ chế tích chập theo cách của S4 được nữa.
- Do đó, tác giả Gu và Dao (2024) đề xuất giải quyết thông qua Selective Scan.

- Selective Scan bao gồm ba kỹ thuật cổ điển: kernel fusion, parallel scan và recomputation.
- Quan sát của tác giả:
    + Recurrent computation: *O(BLDN)* FLOPs.
    + Convolutional computation: *O(BLD log(L))* FLOPs.
    + Với chuỗi quá dài và chiều latent state N không quá lớn, chế độ recurrent có thể dùng ít FLOPs hơn.
    + Thách thức: đặc tính tuần tự của hồi quy và yêu cầu cao về bộ nhớ (giải quyết vấn đề bộ nhớ qua "not actually materialize the full state *h*")*.*

b) Miêu tả thuật toán:

- **Cần thêm kiến thức về phần cứng, kiến trúc máy tính.**

The main idea is to leverage properties of modern accelerators (GPUs) to materialize the state h only in more efficient levels of the memory hierarchy. In particular, most operations (except matrix multiplication) are bounded by memory bandwidth (Dao, Fu, Ermon, et al. 2022; Ivanov et al. 2021; Williams, Waterman, and Patterson 2009). This includes our scan operation, and we use kernel fusion to reduce the amount of memory IOs, leading to a significant speedup compared to a standard implementation.

Concretely, instead of preparing the scan input $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$ of size $(\mathsf{B}, \mathsf{L}, \mathsf{D}, \mathsf{N})$ in GPU HBM (high-bandwidth memory), we load the SSM parameters $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ directly from slow HBM to fast SRAM, perform the discretization and recurrence in SRAM, and then write the final outputs of size $(\mathsf{B}, \mathsf{L}, \mathsf{D})$ back to HBM.

To avoid the sequential recurrence, we observe that despite not being linear it can still be parallelized with a work-efficient parallel scan algorithm (Blelloch 1990; Martin and Cundy 2018; Smith, Warrington, and Linderman 2023).

Finally, we must also avoid saving the intermediate states, which are necessary for backpropagation. We carefully apply the classic technique of recomputation to reduce the memory requirements: the intermediate states are not stored but recomputed in the backward pass when the inputs are loaded from HBM to SRAM. As a result, the fused selective scan layer has the same memory requirements as an optimized transformer implementation with FlashAttention.

Details of the fused kernel and recomputation are in Appendix D. The full Selective SSM layer and algorithm is illustrated in Figure 1.

- Nhìn sơ lược (xem thêm ở ***Appendix D*** của paper Mamba và video của AI Vietnam, 2024):
    + ***Xử lý*** các tác vụ khác nhau ***ở phần cứng phù hợp*** (GPU HBM, SRAM).
    + Thuật toán ***parallel scan***.
    + Hạn chế lưu các trạng thái chuyển tiếp (intermediate states) vào bộ nhớ lúc huấn luyện bằng việc **tính toán lại** các trạng thái trên khi backward.

V. Các lý thuyết khác:

- Controllability & Observability
- Ổn định (Stability) và Lyapunov

B. Setup môi trường cho Mamba:

1. **Sử dụng WSL-Ubuntu cho Windows:** đây là yêu cầu bắt buộc (những nền tảng như Kaggle, Google Colab đã sử dụng Linux sẵn).

2. **Chạy lệnh sau để cài CUDA toolkit:** yêu cầu bởi thư viện Mamba-SSM.
   ```
   sudo apt install nvidia-cuda-toolkit
   ```
3. **Cài đặt thư viện Mamba-SSM với Causal Convolution 1D:** (sẽ không chạy trên Kaggle, cần thêm --no-build-isolation)
   ```
   pip install mamba-ssm[causal-conv1d]
   ```

C. Thực hành với mô hình Mamba

Chúng ta sẽ áp dụng mô hình Mamba cho tác vụ xử lý chuỗi thời gian để test mô hình. (Kaggle)

Tài liệu tham khảo:

Kreyszig, E., Kreyszig, H., & Norminton, E. J. (2011). Advanced engineering mathematics (10th ed.). Wiley.

Gu, A., Goel, J., Ré, C. (2022). Efficiently Modeling Long Sequences with Structured State Space. DOI: arXiv:2111.00396v3

Gu, A., Dao, T. (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. DOI: arXiv:2312.00752v2

Sherstinsky, A. (2023). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. Physica D: Nonlinear Phenomena, 404. DOI: arXiv:1808.03314v10

Appendix:

Appendix [1]:

🧠 Biểu diễn chung của State Space Model (SSM)

Một hệ **tuyến tính bất biến theo thời gian** (Linear Time-Invariant - LTI) có thể được viết như sau:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) & \text{(phương trình trạng thái)} \\ y(t) = Cx(t) + Du(t) & \text{(phương trình đầu ra)} \end{cases}$$

Trong đó:

- $x(t) \in \mathbb{R}^n$: vector trạng thái tại thời điểm t
- $u(t) \in \mathbb{R}^m$: đầu vào (input)
- $y(t) \in \mathbb{R}^p$: đầu ra (output)
- $A \in \mathbb{R}^{n \times n}$: ma trận động học (state transition)
- $B \in \mathbb{R}^{n \times m}$: ma trận điều khiển đầu vào
- $C \in \mathbb{R}^{p \times n}$: ma trận quan sát (observation)
- $D \in \mathbb{R}^{p \times m}$: ma trận truyền trực tiếp từ input sang output