



大连理工大学

DALIAN UNIVERSITY OF TECHNOLOGY

# 从SSM到Mamba的溯源和 相关工作的思考

汇报人：曹逸飞

师从刘宇教授

2024.12.4



# 汇报目录

- 01 ➤ 为什么依然需要SSM
- 02 ➤ 什么是SSM
- 03 ➤ 从SSM到S4再到S6
- 04 ➤ 基于Mamba的一系列工作



大连理工大学

DALIAN UNIVERSITY OF TECHNOLOGY

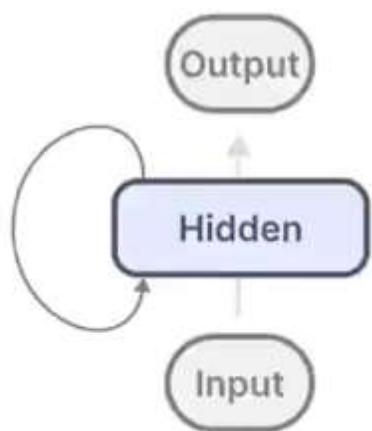
1

# 为什么依然需要SSM

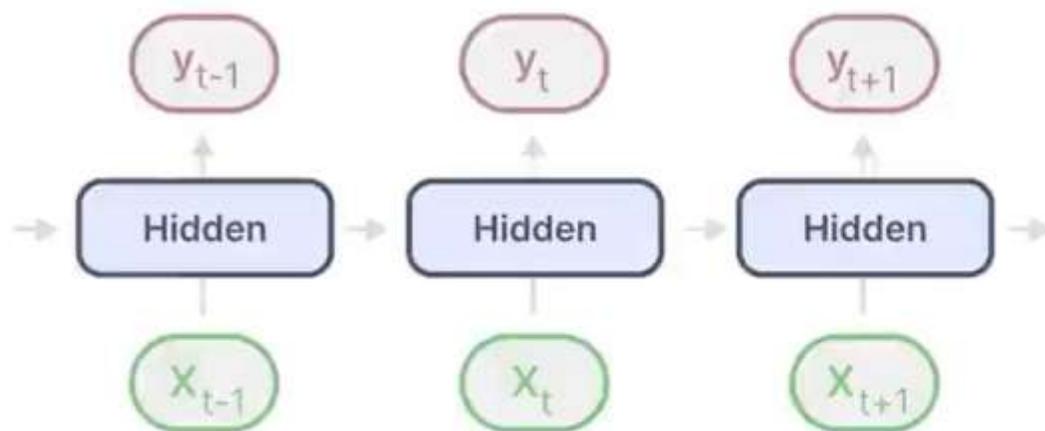
# 01 为什么依然需要SSM

## 1. RNN优势及其问题:

- 线性空间占用，串行导致龟速的训练过程，但能够进行自然且快速地执行推理
  - 训练过程的梯度反向传播被迫逐个通过各个单元，更新参数
  - 推理过程不需要受这一问题影响，因此RNN受益于其较为简单的结构设计，获得了相对快的推理速度
- 遗忘问题，无法有效处理长程依赖关系——即便有了LSTM和GRU等门控的设计，也只是延缓遗忘造成严重影响的位置。
  - 面对长序列仍然无能为力



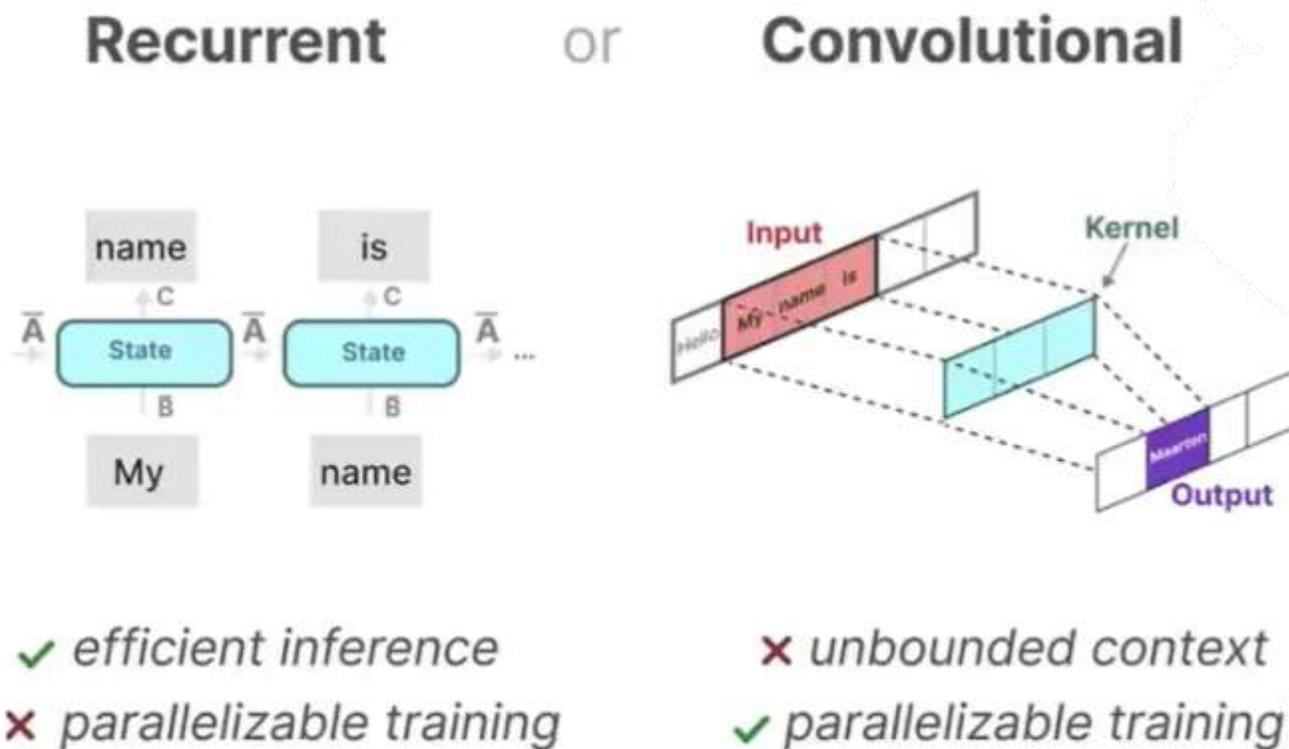
RNN



RNN  
(Unfolded)

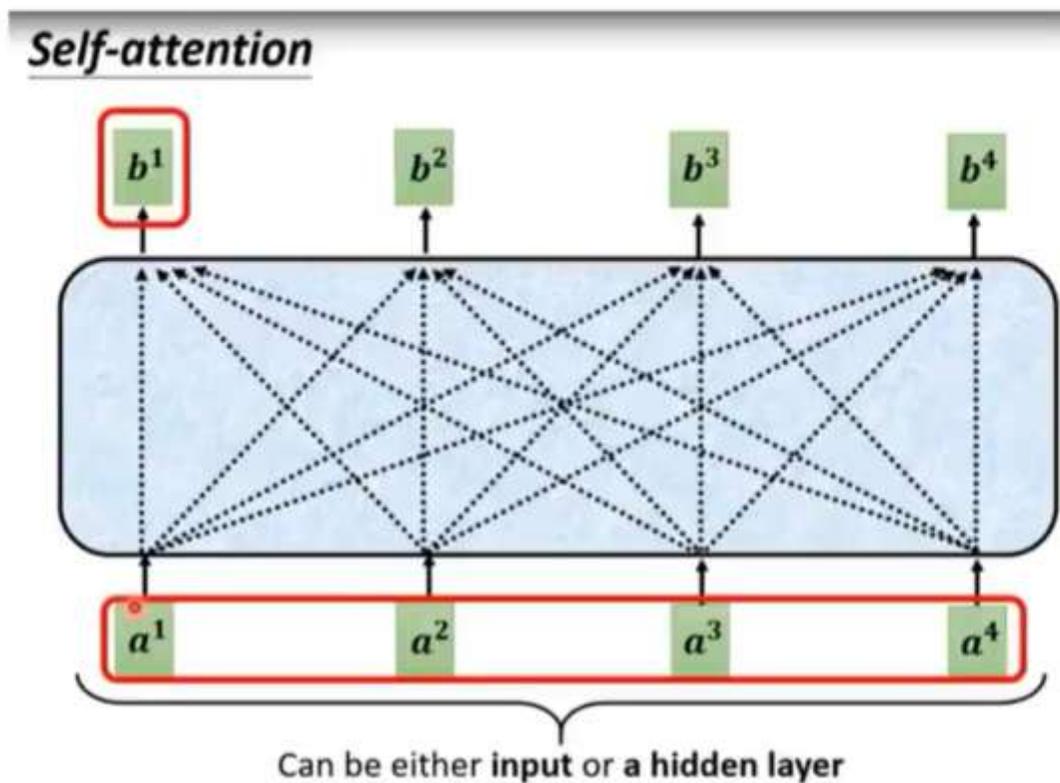
## 2. CNN优势及其问题:

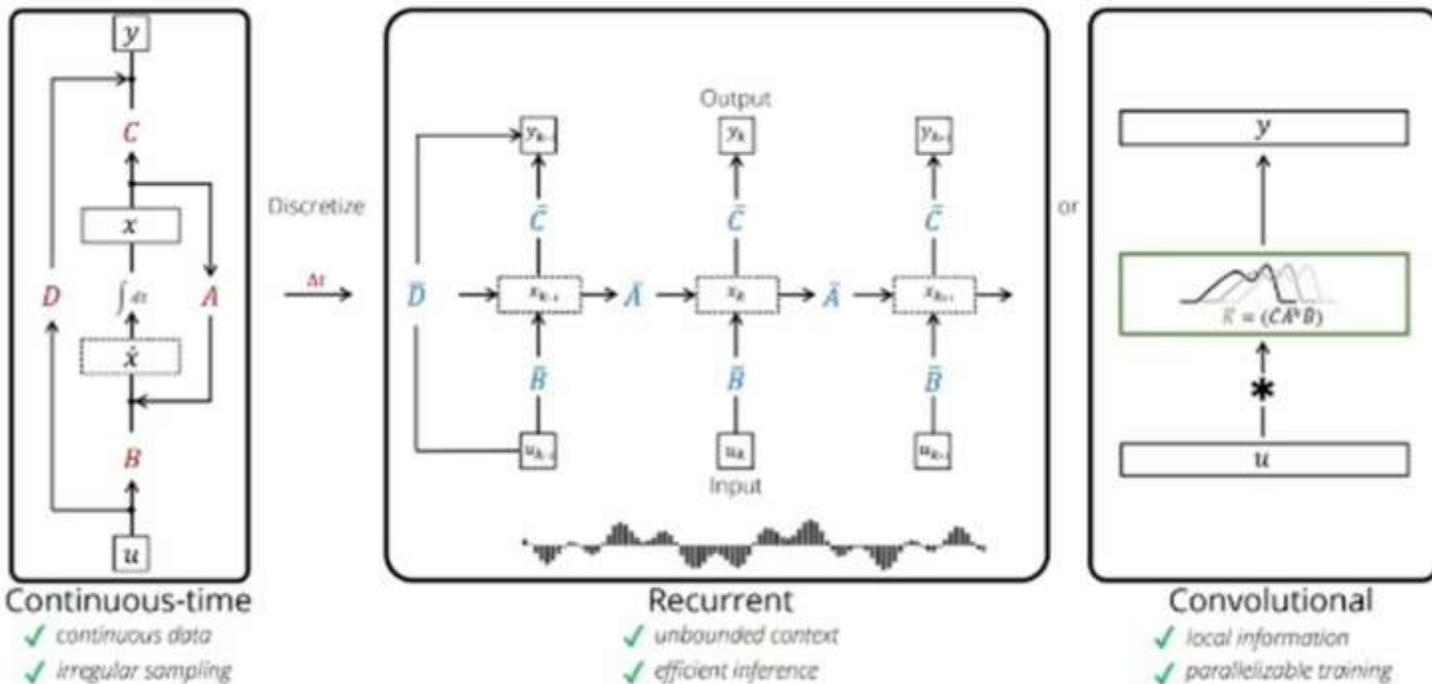
- 线性空间占用，可以并行训练
- 其结构主要关注局部特征，容易忽视全局特征
  - 卷积核主要关注当前元素附近的元素，不能直接使当前元素关注长程的其他元素
- 没有推理方面的优化，固定的卷积核限制了推理速度
  - 相比于RNN，需要额外用卷积核去乘&加多个元素，其推理速度相对平庸



## 3. Attention优势及其问题:

- 可以并行训练，并且可以良好地捕捉长程依赖关系
- 过大的空间复杂度和时间复杂度 $O(L^2)$
- 复杂的注意力分数计算过程造成其推理速度缓慢





## 为什么还可以看成是卷积?

1. 当前时刻的状态  $h_t$ :  
 $h_t = \bar{A}h_{t-1} + \bar{B}x_t$
2. 前一个时刻的状态  $h_{t-1}$ :  
 $h_{t-1} = \bar{A}h_{t-2} + \bar{B}x_{t-1}$

把后式代入前式之后有

$$h_t = \bar{A}^2 h_{t-2} + \bar{A}\bar{B}x_{t-1} + \bar{B}x_t = \sum_{k=0}^t \bar{A}^k \bar{B}x_{t-k}$$

最右边等号是一股化后的情况, 也就是说  $h_t$  可以表示为所有之前时刻的输入的加权和, 注意这里  $t-k$  的含义, 表示了从当前时刻起一个长度为  $k$  的伸缩窗口。再代入输出方程就有,

$$y_t = Ch_t = C \left( \sum_{k=0}^t \bar{A}^k \bar{B}x_{t-k} \right) = \sum_{k=0}^t C\bar{A}^k \bar{B}x_{t-k}$$

对比下面离散卷积的定义, 通过比较找它们的共同点,

$$(x * h)[t] = \sum_{k=0}^{N-1} x[k]h[t-k]$$

$$\begin{aligned} \triangleright h'(t) &= Ah(t) + Bx(t) \\ \triangleright y(t) &= Ch(t) + Dx(t) \end{aligned}$$

Linear Time Invariant系统  
(线性时不变系统)



Non-Linear Time variant系统  
(非线性时变系统)

本质就是一个RNN化的CNN——Transformer???

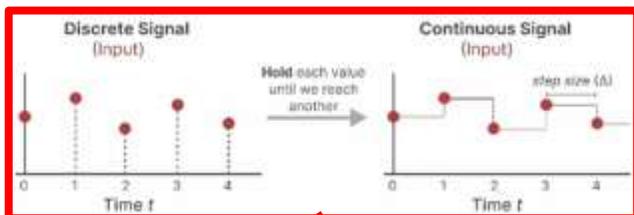
# 02 什么是SSM

## ■ 什么是SSM? ——状态空间模型

### 1. 基本定义

➢ 源于现代控制系统理论。SSM 是用于描述序列在各时间步的状态表示，并根据输入预测其下一个状态的模型。原始理论处理连续函数

- 输入序列  $x(t)$
- 隐状态表示  $h(t)$
- 预测输出序列  $y(t)$



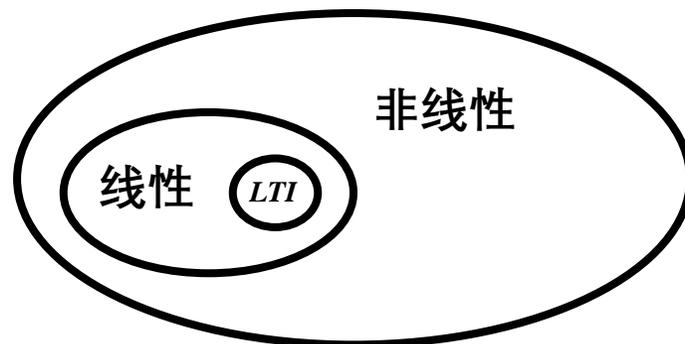
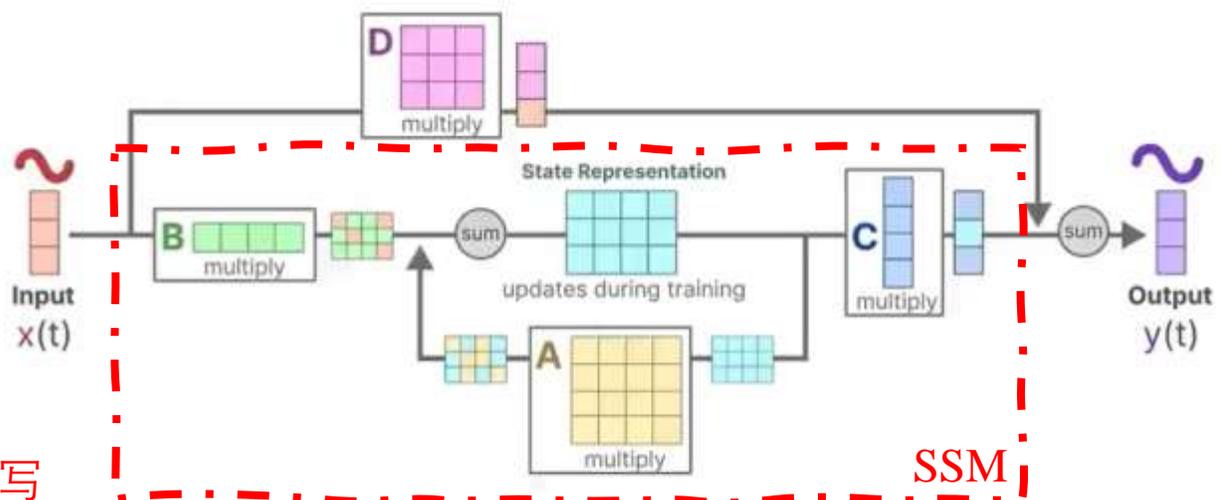
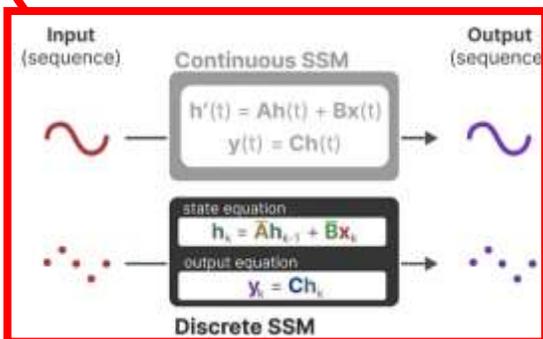
### 2. 公式表示:

- $h'(t) = Ah(t) + Bx(t)$
- $y(t) = Ch(t) + Dx(t)$

### 3. 连续函数离散化——零阶保持

- $h_t = \bar{A}h_{t-1} + \bar{B}x_t, h_0 = \bar{B}x_0$
- $y_t = \bar{C}h_t$

Mamba中D做常数不写



## ■ 什么是SSM? ——状态空间模型 (Structured Space Model)

### 4. 卷积与递归双重属性

$$\begin{aligned}y_2 &= Ch_2 \\ &= C(\bar{A}h_1 + \bar{B}x_2) \\ &= C(\bar{A}(\bar{A}h_0 + \bar{B}x_1) + \bar{B}x_2) \\ &= C(\bar{A}(\bar{A} \cdot \bar{B}x_0 + \bar{B}x_1) + \bar{B}x_2) \\ &= C(\bar{A} \cdot \bar{A} \cdot \bar{B}x_0 + \bar{A} \cdot \bar{B}x_1 + \bar{B}x_2) \\ &= C \cdot \bar{A}^2 \cdot \bar{B}x_0 + C \cdot \bar{A} \cdot \bar{B} \cdot x_1 + C \cdot \bar{B}x_2\end{aligned}$$

➤ 为什么可以看做卷积运算:  $y_k = (\bar{C}\bar{A}^k\bar{B} \quad \bar{C}\bar{A}^{k-1}\bar{B} \quad \dots \quad \bar{C}\bar{A}\bar{B}) \begin{pmatrix} x_0 \\ x_1 \\ \dots \\ x_k \end{pmatrix} \rightarrow$  视为卷积核  $\bar{K} = (\bar{C}\bar{A}^k\bar{B} \quad \bar{C}\bar{A}^{k-1}\bar{B} \quad \dots \quad \bar{C}\bar{A}\bar{B})$ ,  $y = \bar{K}x$

卷积核是通过固定数值的矩阵得到的, 只要能确定ABC, 就可以并行运算

➤ 为什么可以看做递归运算: 如上, 不把括号展开的话, 则构成层层递归过程, 与RNN工作方式一样

➤ 训练时用卷积方式并行, 提高训练效率; 推理时用循环神经网络方式逐个快速生成输出

$$\begin{aligned}h_t &= \bar{A}h_{t-1} + \bar{B}x_{t-1} \\ y_t &= \bar{C}h_t\end{aligned}$$

Kernel

$$\begin{bmatrix} \bar{C}\bar{A}^2\bar{B} & \bar{C}\bar{A}\bar{B} & \bar{C}\bar{B} \end{bmatrix}$$

Input

$(x_k)$



Output

$(y_k)$



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

Multiply

Sum

## From SSM to S4

1. SSM的问题: CNN和RNN的长期依赖捕捉能力都不行, 这导致SSM也有这个问题
2. 解决方案: HIPPO矩阵替代随机初始化的A矩阵

- A矩阵包含着各个时间步隐状态变化的信息, A矩阵左右着信息遗忘过程
- 使用HIPPO矩阵替换矩阵

➢ HIPPO Matrix  $A_{nk}$

$$\begin{cases} (2n+1)^{1/2} (2k+1)^{1/2} & \leftarrow \text{everything below the diagonal} \\ n+1 & \leftarrow \text{the diagonal} \\ 0 & \leftarrow \text{everything above the diagonal} \end{cases}$$

- 相比于随机初始化一个A, 使用HIPPO可以有效缓解遗忘问题
- 为了避免HIPPO本身  $N^2$  的尺寸带来的过大运算量, 利用矩阵分解, 使用低秩矩阵表示HIPPO:

$$A = V \Lambda V^* - P Q^T = V (\Lambda - (V^* P) (V^* Q)^*) V^* \quad P, Q \text{ 长度均为 } N$$

### HiPPO Matrix

1	0	0	0
1	2	0	0
1	3	3	0
1	3	5	4

$n$

$k$

3. S4模型: Structured State Space for Sequences

- 序列的结构状态空间, 一种可以有效建模长时序依赖的SSM模型

*2021 S4—efficiently modeling long sequences with structured state spaces*

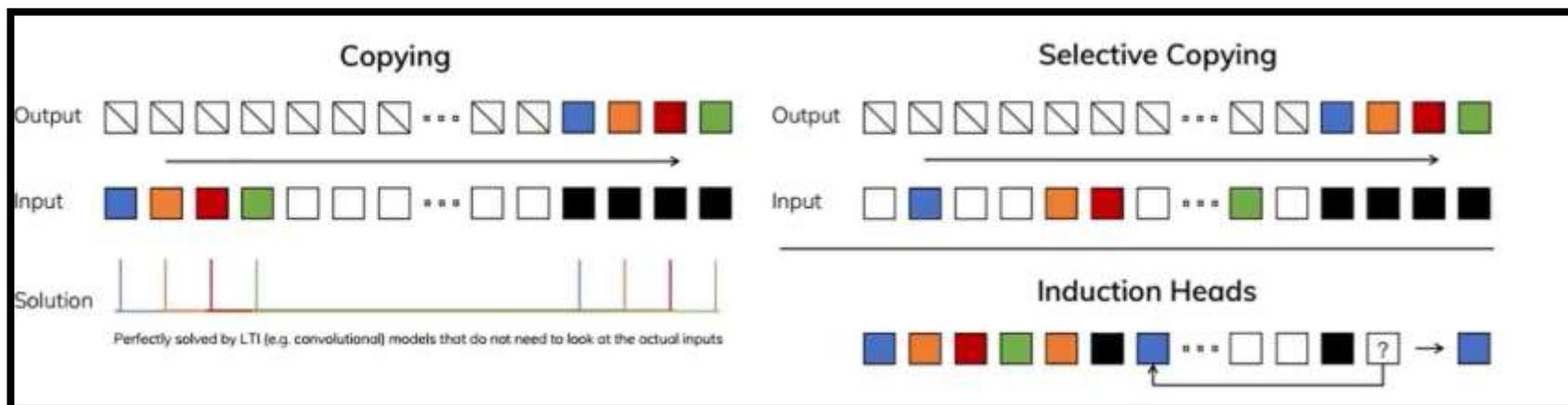
## From S4 to S6 (Mamba)

1. S4与SSM的问题: ABC矩阵, 尤其是最重要的A, 只可能在训练过程中更新自己。一旦训练完成, 无论新的输入是什么, 都会通过完全一样的A, 这导致无法根据输入做针对性推理——无选择性
2. 如果使ABC会根据输入变化后的问题: 无法将SSM转化为标准卷积过程
  - ABC如果能够根据输入的变化而变化, 则可以避免这种无选择性
  - A矩阵和B矩阵参数化后的公式递推: 无法预计算卷积核, 因为不能做到反向传播前的训练的过程保持卷积核的不变

$$\begin{aligned}y_2 &= Ch_2 \\ &= C(\bar{A}_2 h_1 + \bar{B}_2 x_2) \\ &= C(\bar{A}_2(\bar{A}_1 h_0 + \bar{B}_1 x_1) + \bar{B}_2 x_2) \\ &= C(\bar{A}_2(\bar{A}_1(\bar{B}_0 x_0) + \bar{B}_1 x_1) + \bar{B}_2 x_2) \\ &= C\bar{A}_2\bar{A}_1\bar{B}_0 x_0 + C\bar{A}_2\bar{B}_1 x_1 + C\bar{B}_2 x_2\end{aligned}$$

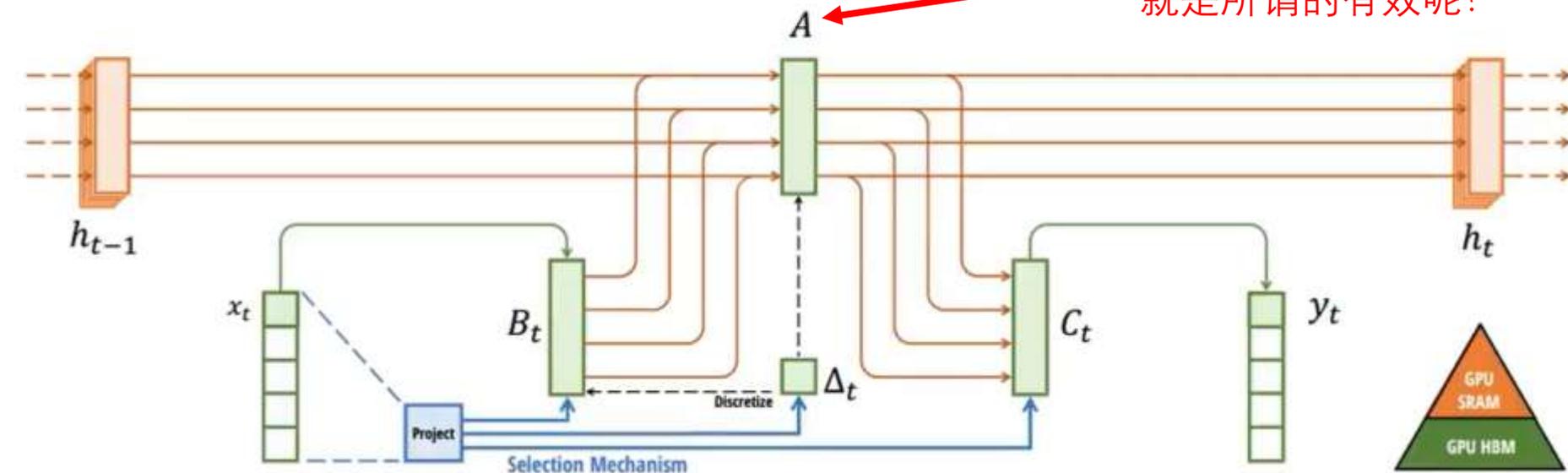
## Mamba: Linear-Time Sequence Modeling with Selective State Spaces

不具备类注意力机制的关注焦点



## Selective State Space Model with Hardware-aware State Expansion

都加了t, 为啥这里不加t呢? 为啥不加t就是所谓的有效呢?



### 1. 参数化矩阵

- 使AB矩阵变为输入数据驱动

### 2. 硬件感知的并行运算算法

- 摆脱卷积运算，单独设计了一个新的可并行运算符
- 减少HBM与SRAM通信次数

### 3. 更简单的SSM结构

## ■ Mamba的具体创新

### 1. 参数化矩阵

- BC变为输入数据驱动的矩阵，尺寸为[B L N]
- $\Delta$ 变为输入数据驱动的矩阵，尺寸为[B L D]
- $\bar{A}, \bar{B}$ 最终变成“针对batch内每一条数据的每个时间步，都有对应矩阵”
- 由于在先前的离散化处理过程中，最终得到的 $\bar{A}, \bar{B}$ 有 $\Delta$ 参与，因此最终运算得到的 $\bar{A}, \bar{B}$ 变成了数据驱动

➢ 不直接把AB参数化成[B L D N]的尺寸，一方面，会造成参数量增大；另一方面，通过上文推导的带有 $\Delta$ 项的运算，因此只需要参数化成如上的尺寸即可实现AB的参数化

➢  $h_{k+1} \in [B, D, N]$

$x_k \in [B, 1, D] \implies A_k h_{k+1} \in [B, D, N]$

$A_k \in [B, D, N] \implies B_k x_k^T \in [B, D, N]$

$B_k \in [B, D, N]$

➢  $h(t_{k+1}) = e^{A^T} h(t_k) + (e^{A^T} - I) \underset{\uparrow}{A}^{-1} \underset{\uparrow}{B} x(t_k)$  记  $\bar{A} = \exp(\Delta A)$ ,  $\bar{B} = (\Delta A)^{-1} (\exp(\Delta A) - I) \Delta B$

➢  $h_t = \bar{A} h_{t-1} + \bar{B} x_{t-1}$

➢  $y_t = \bar{C} h_t$

---

### Algorithm 1 SSM (S4)

---

**Input:**  $x : (B, L, D)$

**Output:**  $y : (B, L, D)$

1:  $A : (D, N) \leftarrow$  Parameter

▷ Represents structured  $N \times N$  matrix

2:  $B : (D, N) \leftarrow$  Parameter

3:  $C : (D, N) \leftarrow$  Parameter

4:  $\Delta : (D) \leftarrow \tau_{\Delta}(\text{Parameter})$

5:  $\bar{A}, \bar{B} : (D, N) \leftarrow \text{discretize}(\Delta, A, B)$

6:  $y \leftarrow \text{SSM}(\bar{A}, \bar{B}, C)(x)$

▷ Time-invariant: recurrence or convolution

7: **return**  $y$

---



---

### Algorithm 2 SSM + Selection (S6)

---

**Input:**  $x : (B, L, D)$

**Output:**  $y : (B, L, D)$

1:  $A : (D, N) \leftarrow$  Parameter

▷ Represents structured  $N \times N$  matrix

2:  $B : (B, L, N) \leftarrow s_B(x)$

3:  $C : (B, L, N) \leftarrow s_C(x)$

4:  $\Delta : (B, L, D) \leftarrow \tau_{\Delta}(\text{Parameter} + s_{\Delta}(x))$

5:  $\bar{A}, \bar{B} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, A, B)$

6:  $y \leftarrow \text{SSM}(\bar{A}, \bar{B}, C)(x)$

▷ Time-varying: recurrence (*scan*) only

7: **return**  $y$

---

## Algorithm 2 SSM + Selection (S6)

**Input:**  $x : (B, L, D)$

**Output:**  $y : (B, L, D)$

1:  $A : (D, N) \leftarrow \text{Parameter}$

▷ Represents structured  $N \times N$  matrix

2:  $B : (B, L, N) \leftarrow s_B(x)$

3:  $C : (B, L, N) \leftarrow s_C(x)$

4:  $\Delta : (B, L, D) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$

5:  $\bar{A}, \bar{B} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, A, B)$

6:  $y \leftarrow \text{SSM}(\bar{A}, \bar{B}, C)(x)$

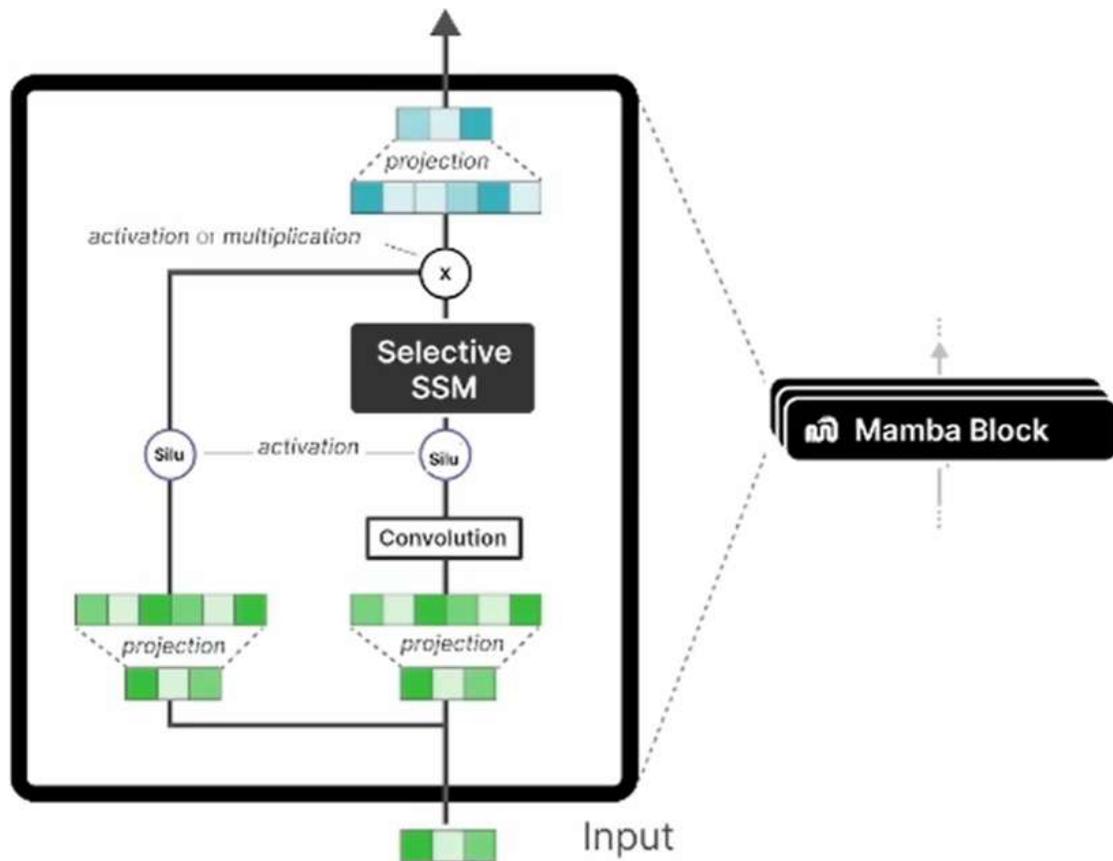
▷ Time-varying: recurrence (*scan*) only

7: **return**  $y$

$$S_C(x) = \text{Linear}_N(x)$$

$$S_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x))$$

$$\tau_\Delta = \text{softplus}$$



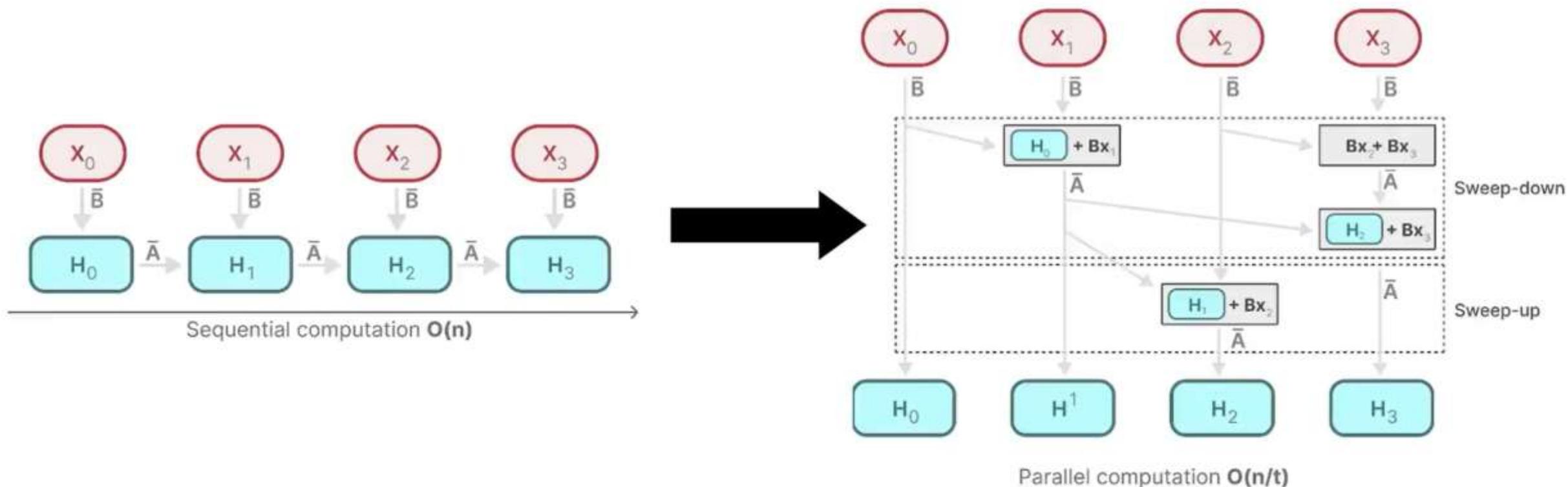
选择性的体现主要就是上面这三个， 因为是根据输入变化的

1. 输入  $x(B, L, D)$ , 由线性层转换为  $B(B, L, N)$ , 另一个线性层转换为  $C(B, L, N)$ , 第三个线性层转换为  $\Delta(B, L, D)$ , 再使用 `softplus` 激活函数处理, 得到  $\Delta(B, L, D)$ 。
2. 系统矩阵  $A(D, N)$  是通过参数进行加载。使用  $\Delta(B, L, D)$  处理  $A$  和  $B$ :  $\bar{A}: \text{torch.einsum}('bld, dn \rightarrow bldn', \Delta, A)$ ;  $\bar{B}: \text{torch.einsum}('bld, bln \rightarrow bldn', \Delta, B)$  得到  $\bar{B}(B, L, D, N)$
3. 计算状态  $h$ : 加载上一时刻  $h(B, L, D, N)$ , 并计算新的状态  $h_{new}$ :  $\text{torch.einsum}('bldn, bldn \rightarrow bldn', \bar{A}, h) + \text{rearrange}(x, 'bld \rightarrow bld 1') \times \bar{B}$  对应位置相乘。
4. 计算输出  $y$ :  $\text{torch.einsum}('bln, bldn \rightarrow bld', C, h_{new})$  这里  $y$  的维度就与  $x$  维度一样了。

## ■ Mamba的具体创新

### 2. 硬件感知算法

- 并行化——选择性扫描算法 (selective scan algorithm)
- 放弃用卷积来描述SSM，而是定义了一种新的“加”运算，在并行计算中，“连加”操作是可并行的：



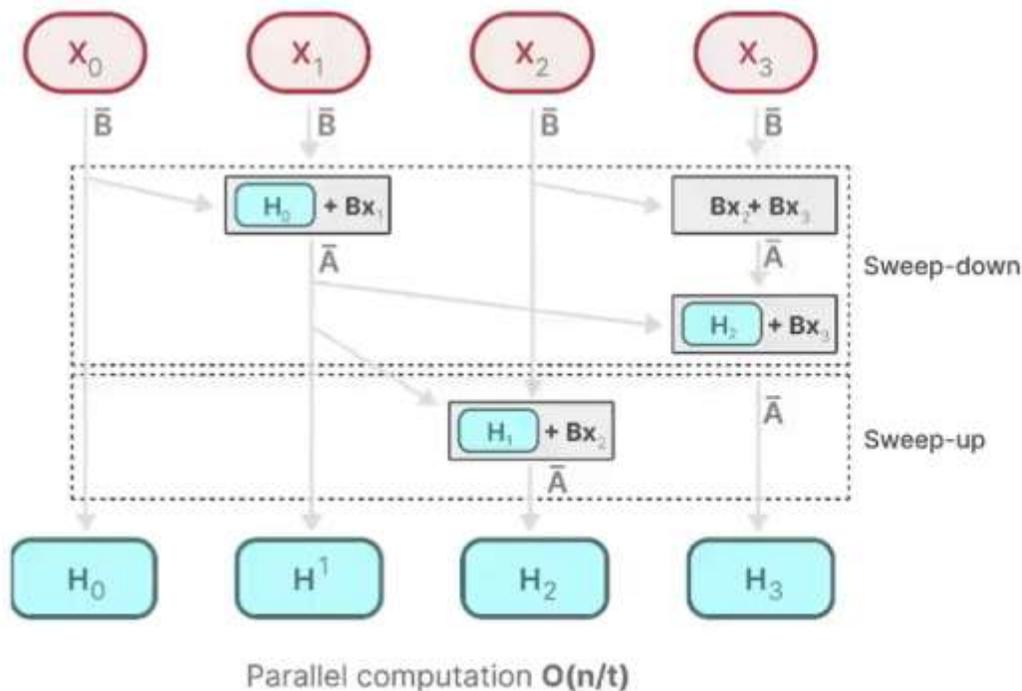
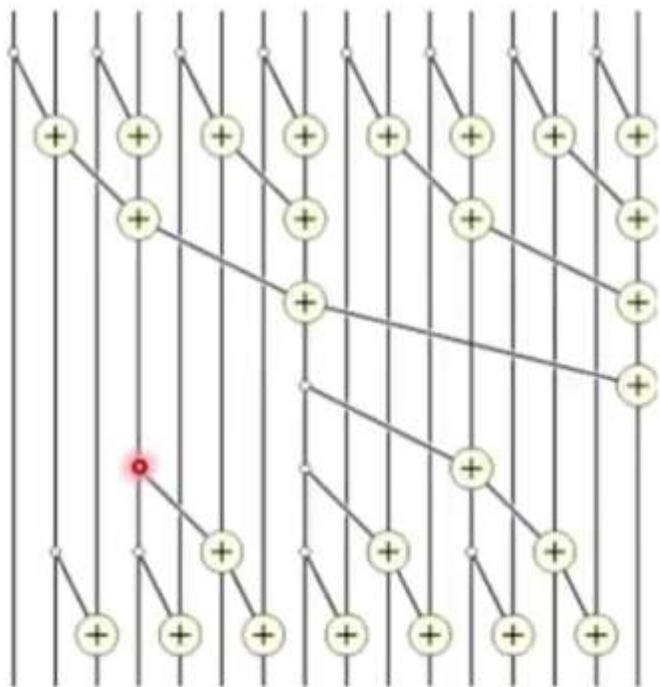
## ■ Mamba的具体创新

### 2. 硬件感知算法

#### ➢ 并行化——选择性扫描算法 (selective scan algorithm)

- 放弃用卷积来描述SSM，而是定义了一种新的“加”运算，在并行计算中，“连加”操作是可并行的：
- 假设运算操作的顺序与关联矩阵  $A$  无关，会发现每个  $x_t$  乘的矩阵都是源于  $x_t$  本身的（矩阵  $BC$  是  $x$  通过线性层得到的）
- 定义新的运算过程： $(A_t, B_t x_t) \oplus (A_{t+1}, B_{t+1} x_{t+1}) = (A_t A_{t+1}, A_{t+1} B_t x_t + B_{t+1} x_{t+1})$
- 该运算符满足交换律&结合律，取该运算符的第二项结果。虽然不是卷积运算，但是一种并行运算

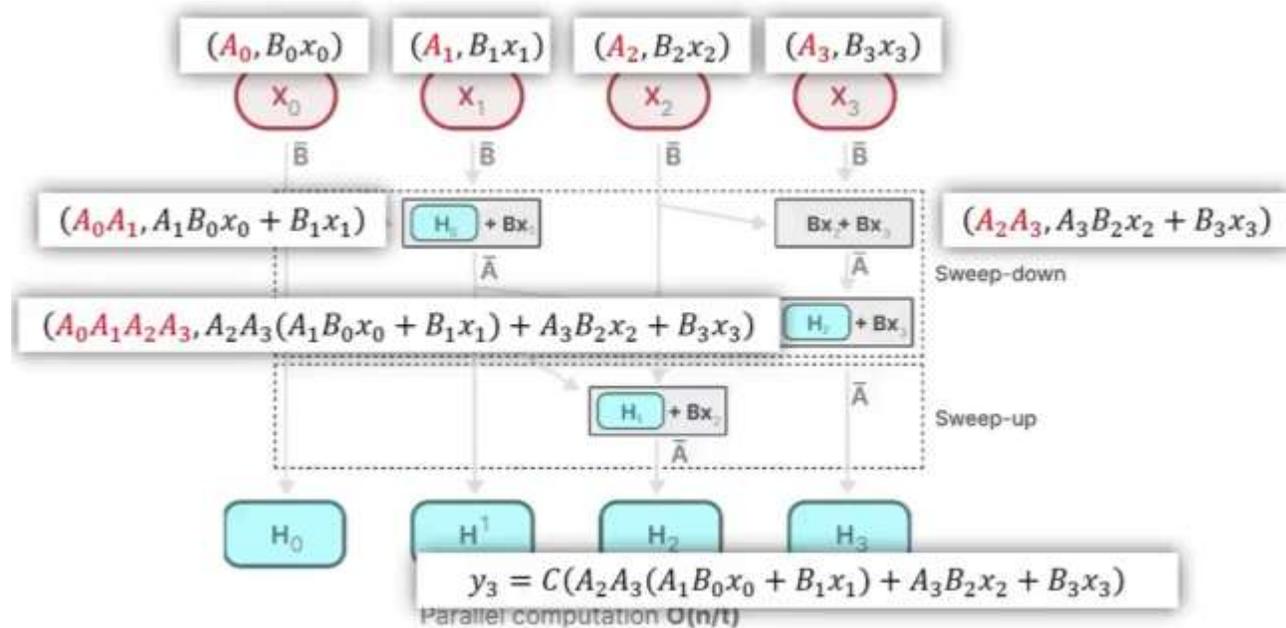
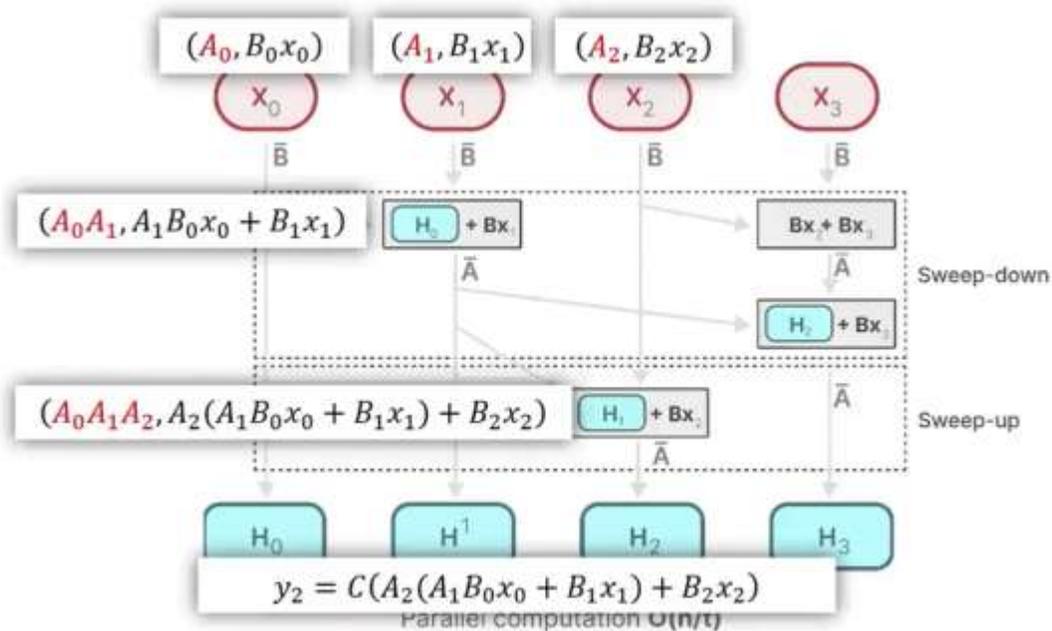
$$C\bar{A}_2\bar{A}_1\bar{B}_0x_0 + C\bar{A}_2\bar{B}_1x_1 + C\bar{B}_2x_2$$



$$(A_t, B_t x_t) \oplus (A_{t+1}, B_{t+1} x_{t+1}) = (A_t A_{t+1}, A_{t+1} B_t x_t + B_{t+1} x_{t+1})$$

$$\begin{aligned} y_2 &= C h_2 \\ &= C(\bar{A}_2 h_1 + \bar{B}_2 x_2) \\ &= C(\bar{A}_2(\bar{A}_1 h_0 + \bar{B}_1 x_1) + \bar{B}_2 x_2) \\ &= C(\bar{A}_2(\bar{A}_1(\bar{B}_0 x_0) + \bar{B}_1 x_1) + \bar{B}_2 x_2) \\ &= C\bar{A}_2\bar{A}_1\bar{B}_0 x_0 + C\bar{A}_2\bar{B}_1 x_1 + C\bar{B}_2 x_2 \end{aligned}$$

那y3呢?

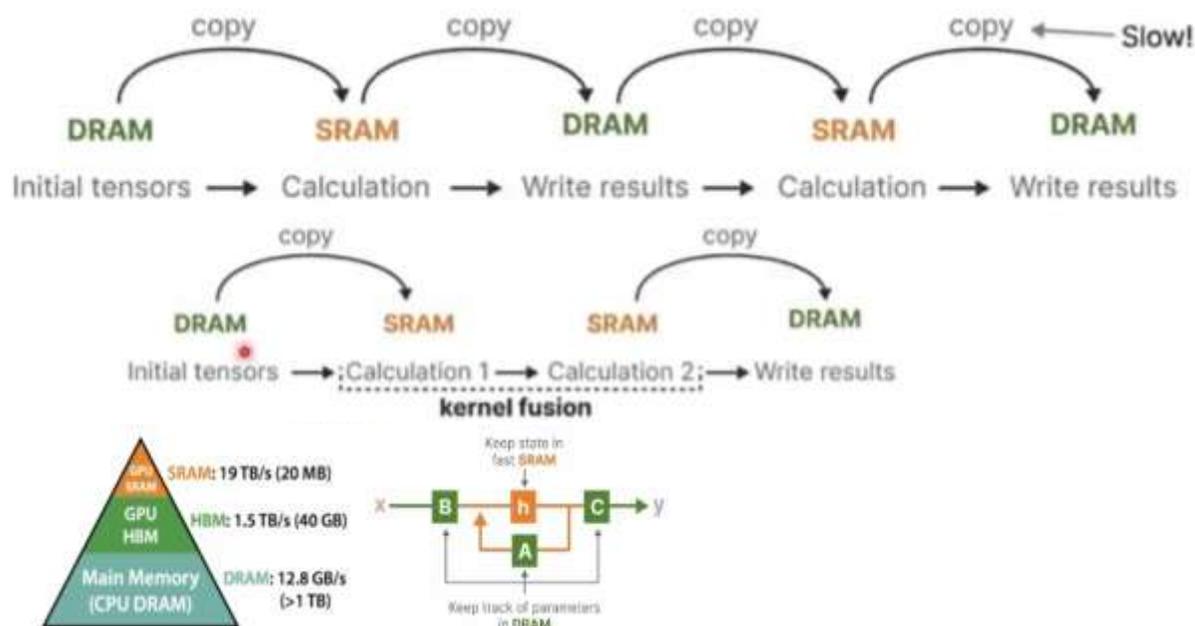


## ■ Mamba的具体创新

计算中的瓶颈在于数据在内存中的移动，往往并不是计算本身

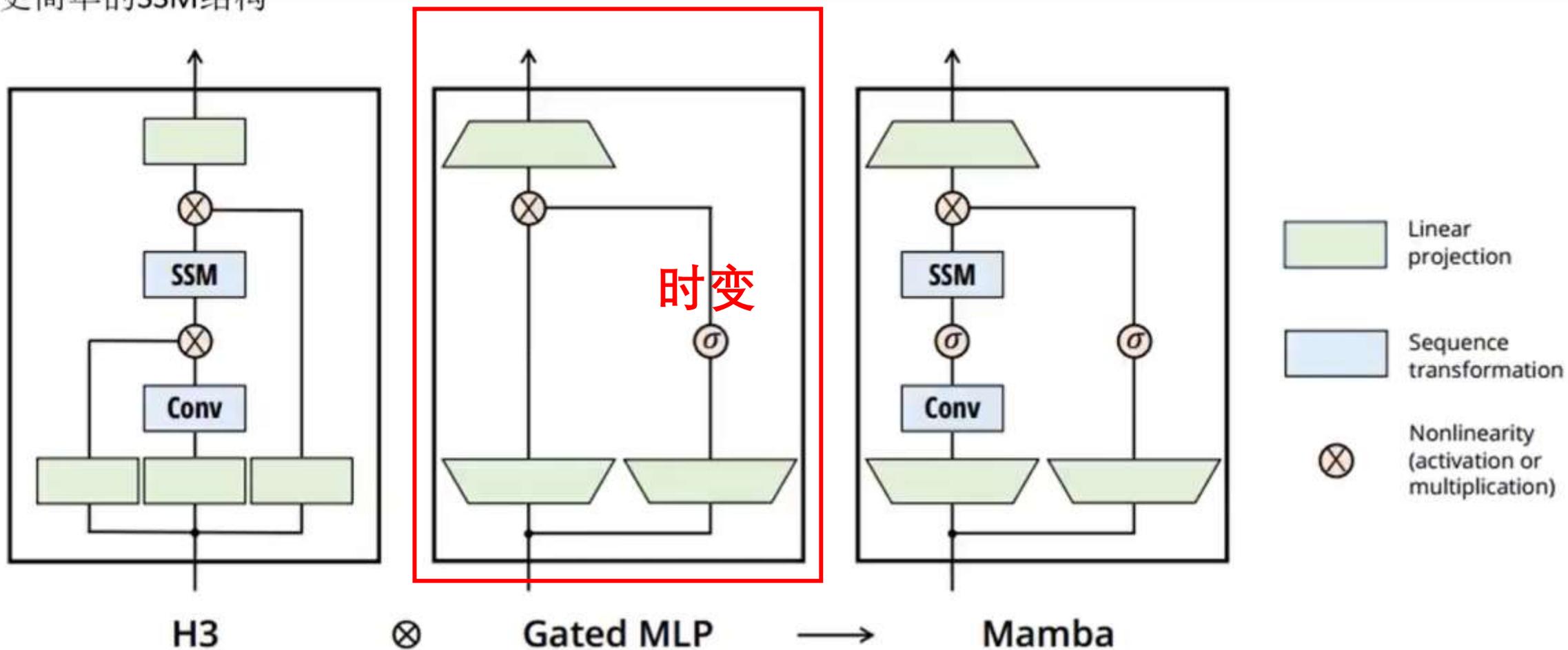
### 2. 硬件感知算法

- 并行化——选择性扫描算法 (selective scan algorithm)
- 利用SSM本身显存占用小的优势，争取模型和运算过程全部放在SRAM完成；相比之下，Transformer显存占用过大，无法完成这种事情
  - HBM：显卡的高带宽内存，提供了比传统的 GDDR 更高的带宽，更低的功耗。当然，相比于SRAM，HBM仍是“低速大容量”的
  - SRAM：显卡的高速缓存区，读取速度非常快
  - Transformer 仅注意力层可能就需要把模型各个模块分批次从HBM加载到SRAM去计算，一个模块算完了就从SRAM取出来，再加载下一个模块如，先算QKV，再算注意力分数，注意力分数再与输入相乘
  - SSM的参数（原始的 $A, B, C, \Delta$  会被直接加载到SRAM，在SRAM里计算 $\bar{A}, \bar{B}$  及后续操作，一步直接得到输出，从SRAM写回HBM）

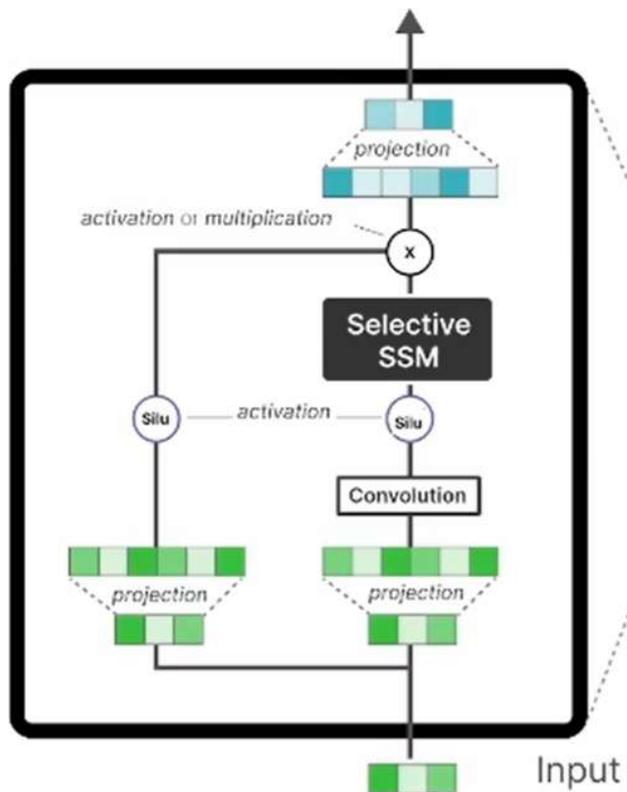


## ■ Mamba的具体创新

### 3. 更简单的SSM结构

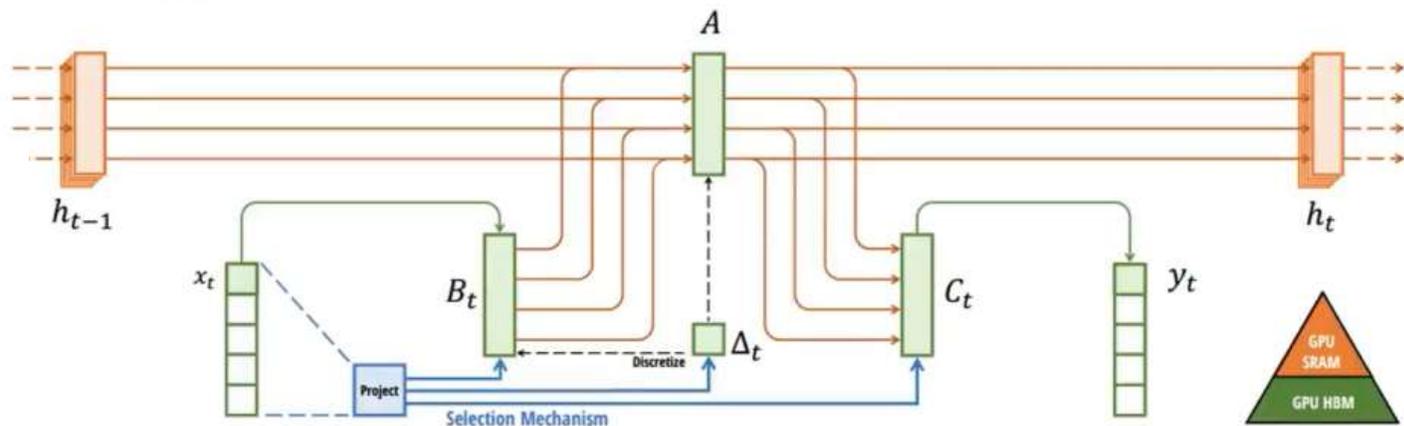


线性投影+卷积操作+SSM+门控



Mamba Block

Selective State Space Model  
with Hardware-aware State Expansion



从粒子运动到流体运动？李指数映射的问题

	Convolution <sup>3</sup>	Recurrence	Attention	S4
Parameters	$LH$	$H^2$	$H^2$	$H^2$
Training	$\tilde{L}H(B + H)$	$BLH^2$	$B(L^2H + LH^2)$	$BH(\tilde{H} + \tilde{L}) + B\tilde{L}H$
Space	$BLH$	$BLH$	$B(L^2 + HL)$	$BLH$
Parallel	Yes	No	Yes	Yes
Inference	$LH^2$	$H^2$	$L^2H + H^2L$	$H^2$

## ICML2024

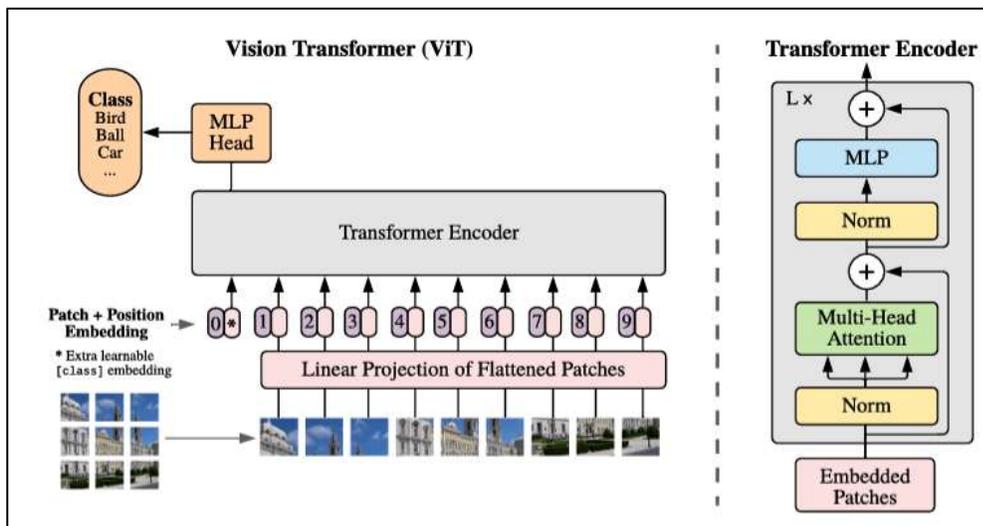
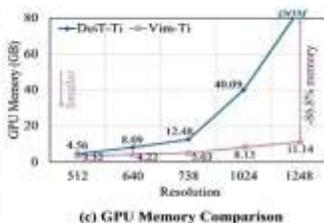
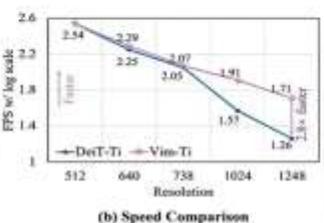
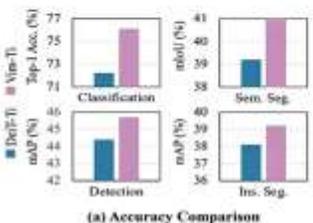
### Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model

Lianghui Z<sup>1\*</sup>, Bencheng Liao<sup>1\*</sup>, Qian Zhang<sup>2</sup>, Xinlong Wang<sup>3</sup>, Wenyu Liu<sup>1</sup>, Xinggang Wang<sup>1</sup> 

<sup>1</sup> Huazhong University of Science and Technology

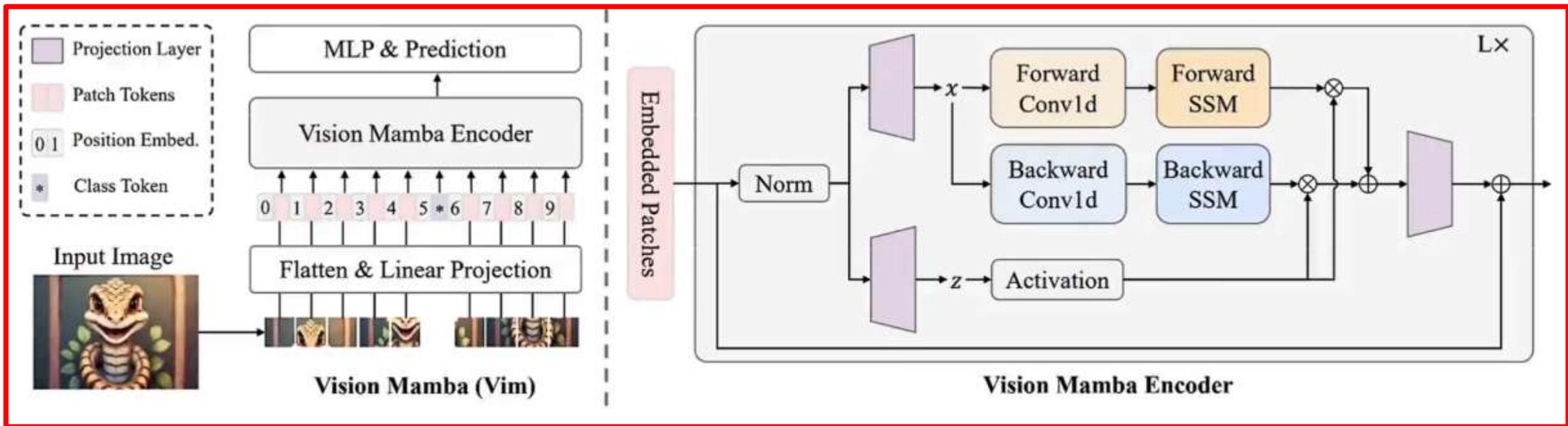
<sup>2</sup> Horizon Robotics <sup>3</sup> Beijing Academy of Artificial Intelligence

Code & Models: [hustvl/vim](https://github.com/hustvl/vim)



- 共同点
  - 切分patch
  - 1-D序列
  - 位置编码
  - 框架结构

- 不同点
  - Patch处理
  - Clstoken位置



## Algorithm 1 Vim Block Process

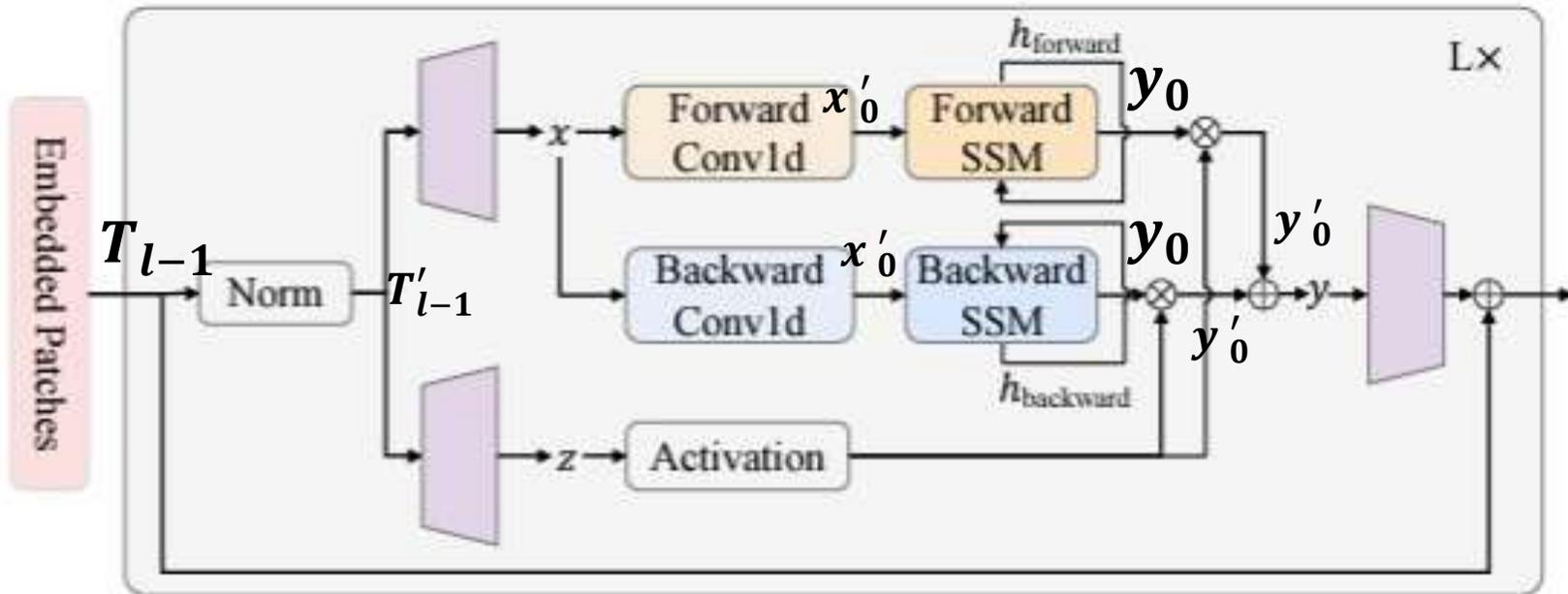
**Require:** token sequence  $T_{l-1} : (B, M, D)$

**Ensure:** token sequence  $T_l : (B, M, D)$

```

1: /* normalize the input sequence  $T'_{l-1}$  */
2:  $T'_{l-1} : (B, M, D) \leftarrow \text{Norm}(T_{l-1})$ 
3:  $x : (B, M, E) \leftarrow \text{Linear}^x(T'_{l-1})$ 
4:  $z : (B, M, E) \leftarrow \text{Linear}^z(T'_{l-1})$ 
5: /* process with different direction */
6: for  $o$  in {forward, backward} do
7:    $x'_o : (B, M, E) \leftarrow \text{SiLU}(\text{Conv1d}_o(x))$ 
8:    $B_o : (B, M, N) \leftarrow \text{Linear}^B_o(x'_o)$ 
9:    $C_o : (B, M, N) \leftarrow \text{Linear}^C_o(x'_o)$ 
10:  /* softplus ensures positive  $\Delta_o$  */
11:   $\Delta_o : (B, M, E) \leftarrow \log(1 + \exp(\text{Linear}^\Delta_o(x'_o) + \text{Parameter}^\Delta_o))$ 
12:  /* shape of  $\text{Parameter}^\Delta_o$  is  $(E, N)$  */
13:   $\overline{A}_o : (B, M, E, N) \leftarrow \Delta_o \otimes \text{Parameter}^\Delta_o$ 
14:   $\overline{B}_o : (B, M, E, N) \leftarrow \Delta_o \otimes B_o$ 
15:  /* initialize  $h_o$  and  $y_o$  with 0 */
16:   $h_o : (B, E, N) \leftarrow \text{zeros}(B, E, N)$ 
17:   $y_o : (B, M, E) \leftarrow \text{zeros}(B, M, E)$ 
18:  /* SSM recurrent */
19:  for  $i$  in {0, ..., M-1} do
20:     $h_o = \overline{A}_o[:, i, :, :] \odot h_o + \overline{B}_o[:, i, :, :] \odot x'_o[:, i, :, \text{None}]$ 
21:     $y_o[:, i, :] = h_o \otimes C_o[:, i, :]$ 
22:  end for
23: end for
24: /* get gated y */
25:  $y'_{forward} : (B, M, E) \leftarrow y_{forward} \odot \text{SiLU}(z)$ 
26:  $y'_{backward} : (B, M, E) \leftarrow y_{backward} \odot \text{SiLU}(z)$ 
27: /* residual connection */
28:  $T_l : (B, M, D) \leftarrow \text{Linear}^T(y'_{forward} + y'_{backward}) + T_{l-1}$ 
29: Return:  $T_l$ 

```



Vision Mamba Encoder

**P:** patchsize(16)

**L:** ViM Block数量(24)

**B:** Batchsize

**M:**输入的序列的长度，即一个图像拆分为多少个token

**D:**192 token的通道数

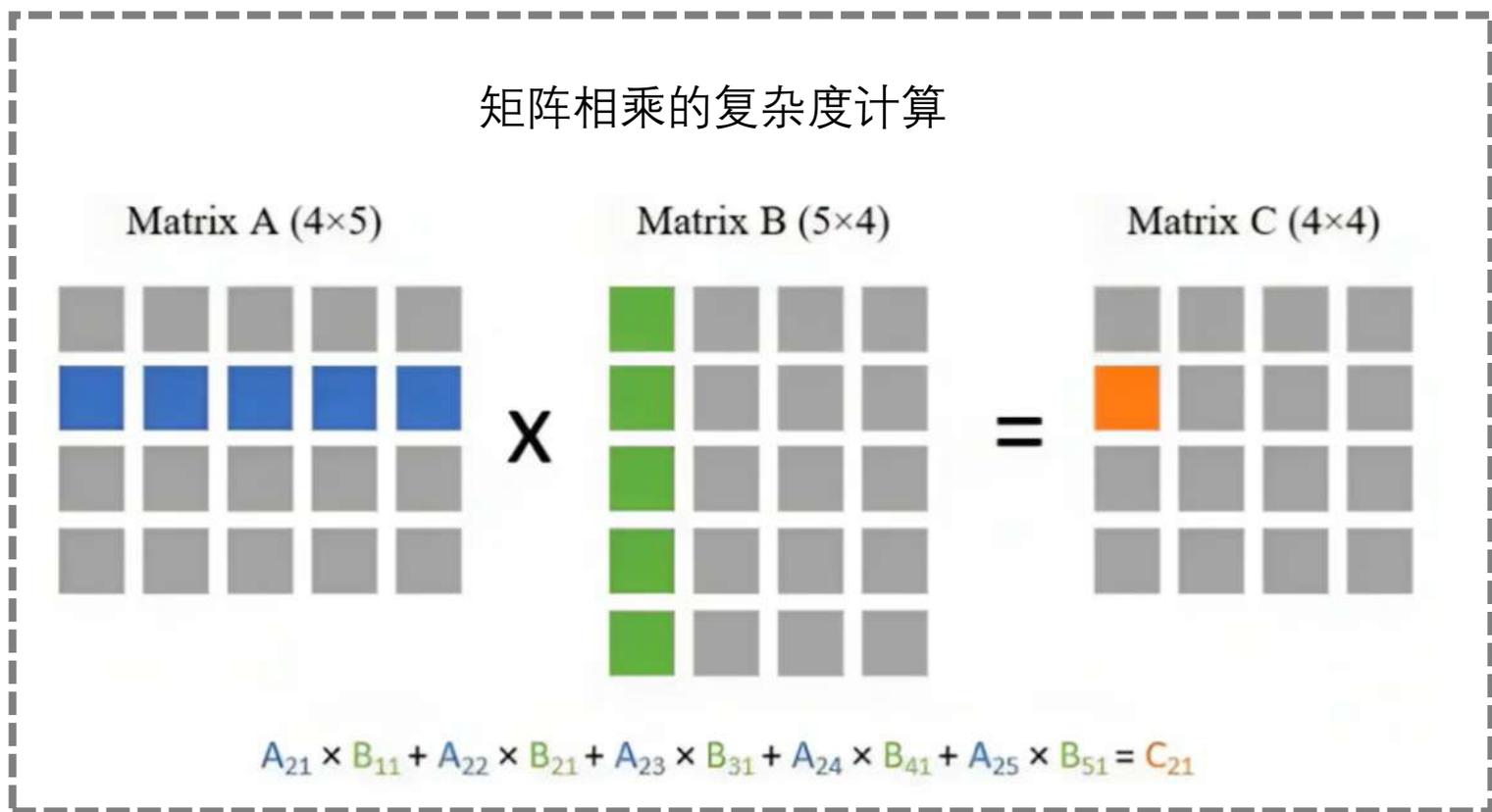
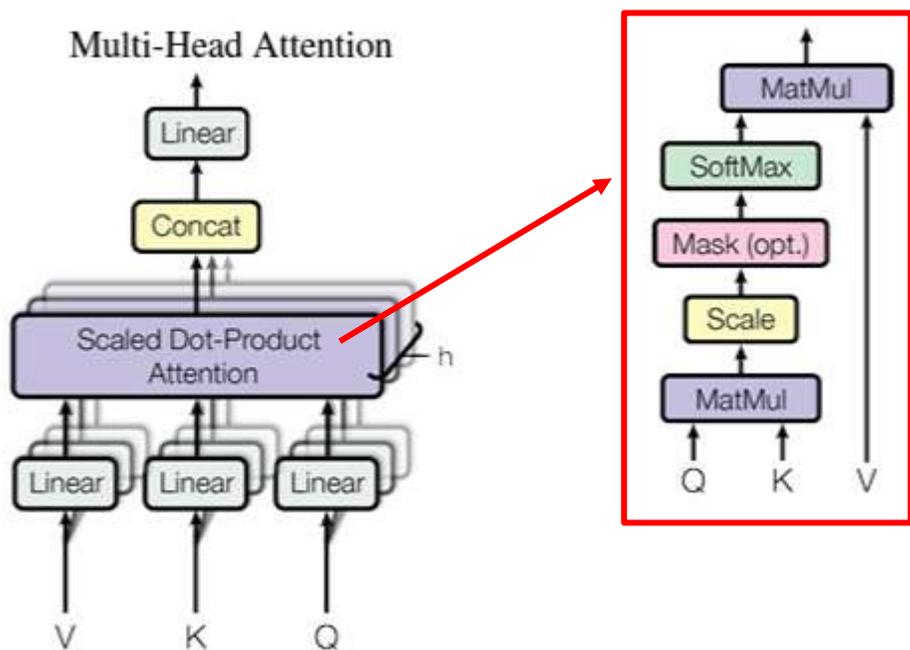
**N:** 16 特征方程的阶数/SSM的维度

**E:** 384 or 768 D扩展的通道数

## 复杂度和计算效率

$$\Omega(\text{self-attention}) = 4MD^2 + 2M^2D, \longrightarrow \text{二次方复杂度}$$

$$\Omega(\text{SSM}) = 3M(2D)N + M(2D)N, \longrightarrow \text{一次方复杂度 (适合大尺寸图像)}$$



## 复杂度和计算效率

$$\Omega(\text{self-attention}) = 4MD^2 + 2M^2D, \longrightarrow \text{二次方复杂度}$$

$$\Omega(\text{SSM}) = 3M(2D)N + M(2D)N, \longrightarrow \text{一次方复杂度 (适合大尺寸图像)}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

对于输入的 $x$  ( $B * M * D$ ), 设 $\text{batchsize}=1$ , 即 ( $M * D$ )

Transformer:

- ◆  $X$  ( $M * D$ ) 与  $W_{qkv}$  ( $D * D$ ) 得到 query、key 和 value ( $M * D$ ), 复杂度就是  $MD^2$ , 有三个 QKV 所以是  $3MD^2$
- ◆  $Q$  ( $M * D$ ) 和  $K$  ( $D * M$ ) 的转置相乘得到相似性矩阵 ( $M * M$ ), 复杂度是  $M^2D$
- ◆ 相似矩阵 ( $M * M$ ) 再和  $V$  ( $M * D$ ) 做运算得到矩阵 ( $M * D$ ), 复杂度为  $M^2D$
- ◆ 再缩放之后经过  $\text{linear}$  ( $D * D$ ) 得到最终的结构 ( $M * D$ ), 复杂度为  $MD^2$

因此总复杂度为  $4MD^2 + 2M^2D$

Mamba:

经过线性映射为 ( $M, E$ ) 也就是 ( $M, 2D$ ), 然后转化为 ( $M, 2D, 1$ )

$X(M, 2D, 1) \times \bar{B}(M, 2D, N)$ , 计算复杂度为 ( $M2DN$ )

$h(M, 2D, N) \times \bar{A}(M, 2D, N)$  计算复杂度为 ( $M2DN$ )

$h(M, 2D, N) \times \bar{C}(M, 2D, N)$  计算复杂度为 ( $M2DN$ )

# 04 基于Mamba的工作

Method	image size	#param.	ImageNet top-1 acc.
<b>Convnets</b>			
ResNet-18	224 <sup>2</sup>	12M	69.8
ResNet-50	224 <sup>2</sup>	25M	76.2
ResNet-101	224 <sup>2</sup>	45M	77.4
ResNet-152	224 <sup>2</sup>	60M	78.3
ResNeXt50-32×4d	224 <sup>2</sup>	25M	77.6
RegNetY-4GF	224 <sup>2</sup>	21M	80.0
<b>Transformers</b>			
ViT-B/16	384 <sup>2</sup>	86M	77.9
ViT-L/16	384 <sup>2</sup>	307M	76.5
DeiT-Ti	224 <sup>2</sup>	6M	72.2
DeiT-S	224 <sup>2</sup>	22M	79.8
DeiT-B	224 <sup>2</sup>	86M	81.8
<b>SSMs</b>			
S4ND-ViT-B	224 <sup>2</sup>	89M	80.4
Vim-Ti	224 <sup>2</sup>	7M	76.1
Vim-Ti <sup>†</sup>	224 <sup>2</sup>	7M	78.3 +2.2
Vim-S	224 <sup>2</sup>	26M	80.5
Vim-S <sup>†</sup>	224 <sup>2</sup>	26M	81.6 +1.1

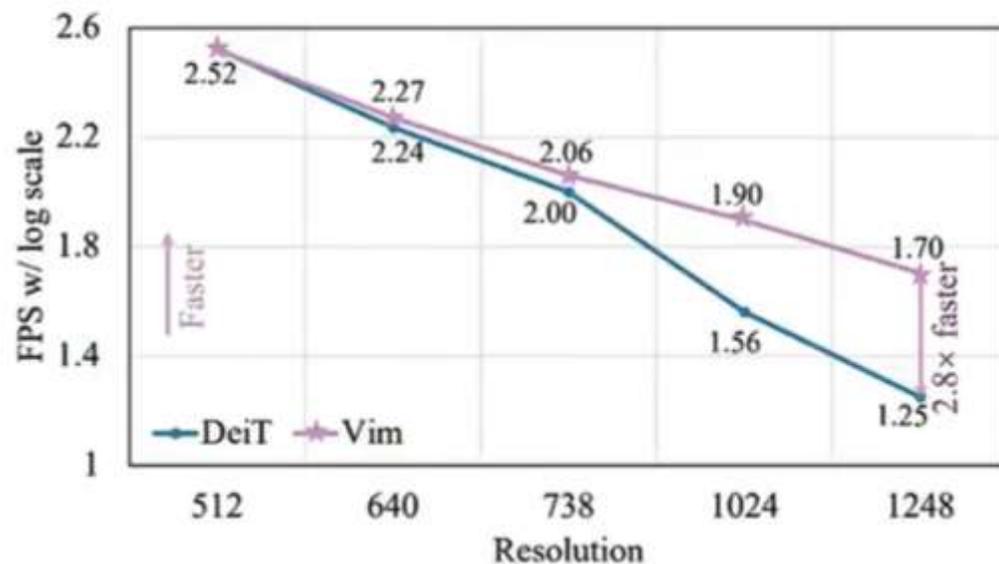
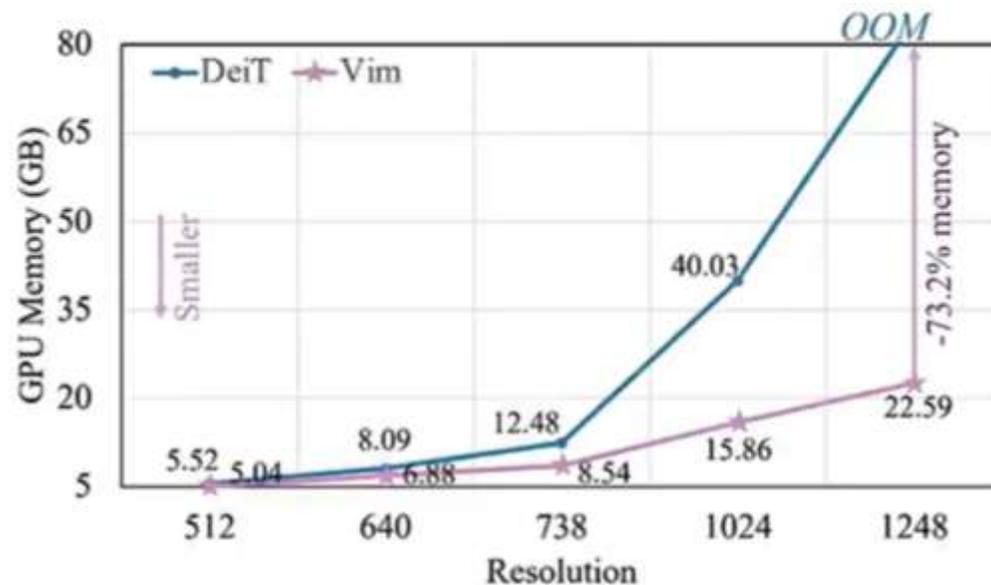


Table 1. Comparison with different backbones on ImageNet-1K validation set. <sup>†</sup> represents the model is fine-tuned with our long sequence setting.

## MambaTrack: A Simple Baseline for Multiple Object Tracking with State Space Model

Changcheng Xiao†

xiaocc612@foxmail.com

National University of Defense Technology  
Changsha, Hunan Province, China

Zhigang Luo

zgluo@nudt.edu.cn

National University of Defense Technology  
Changsha, Hunan Province, China

Qiong Cao†

mathqiong2012@gmail.com

JD Explore Academy  
Beijing, China

Long Lan\*

long.lan@nudt.edu.cn

National University of Defense Technology  
Changsha, Hunan Province, China

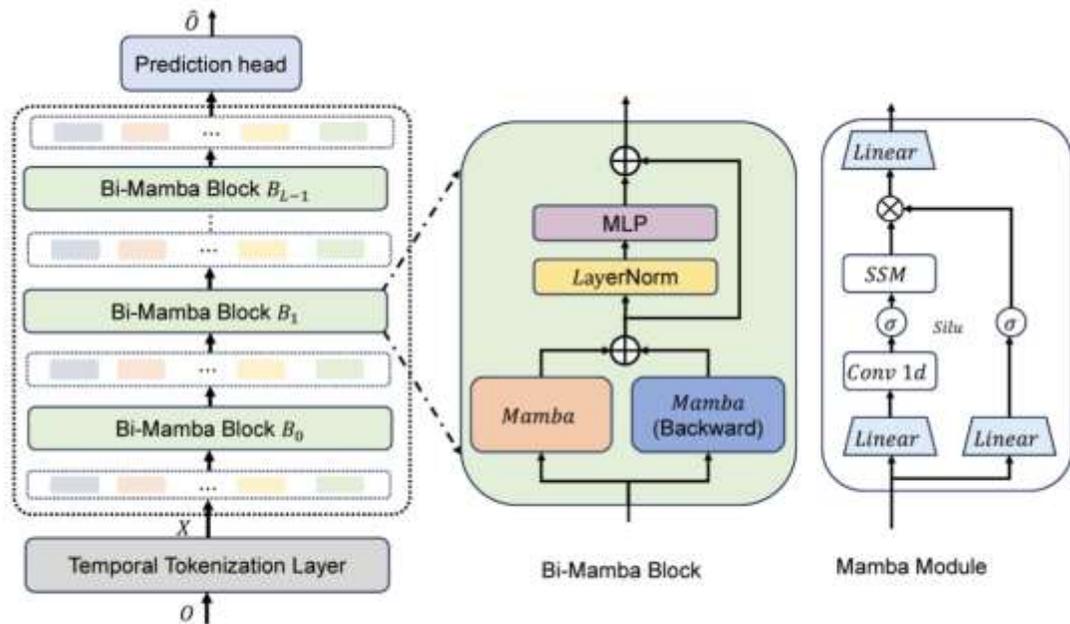
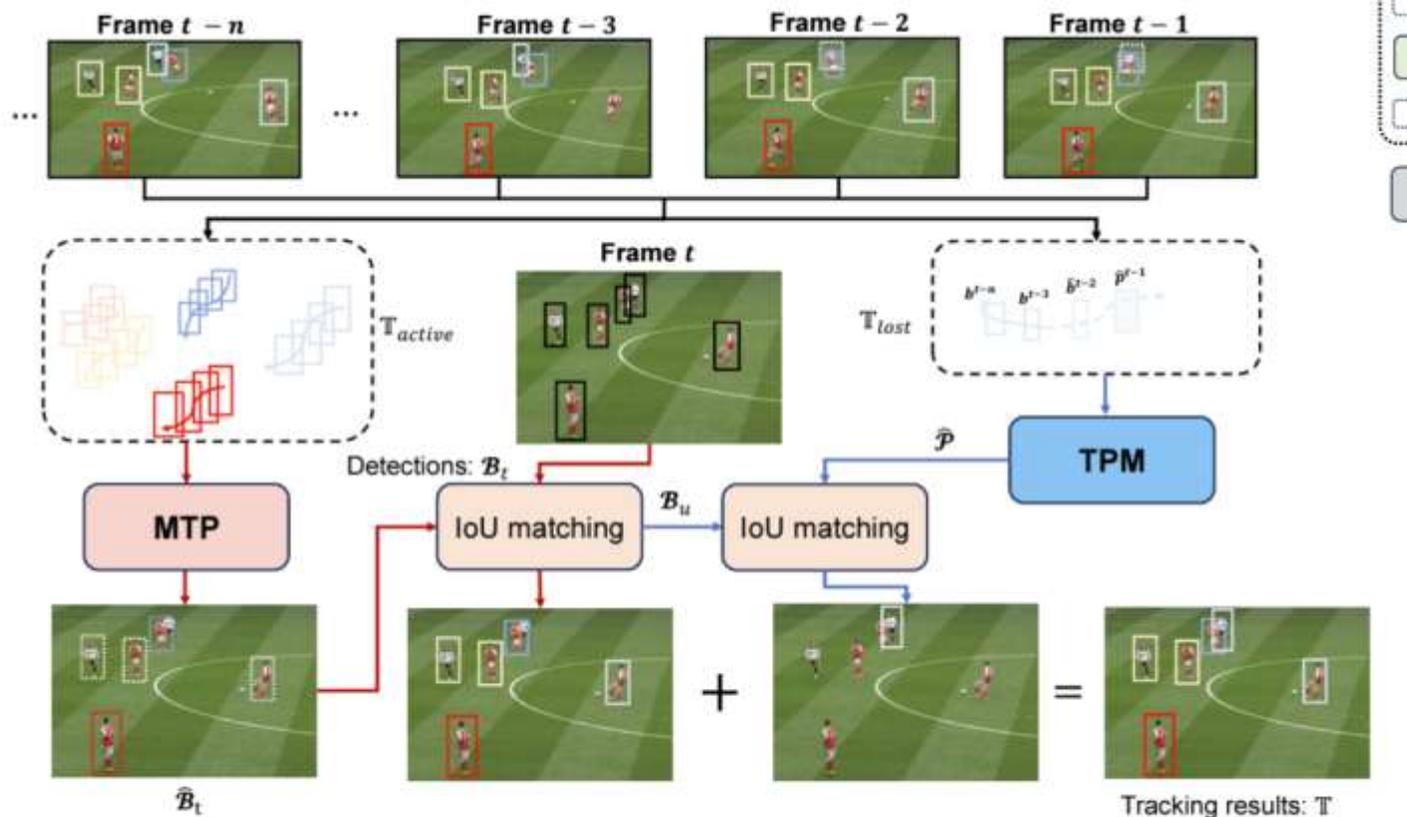


Figure 2: Overview of the proposed Mamba motion predictor.

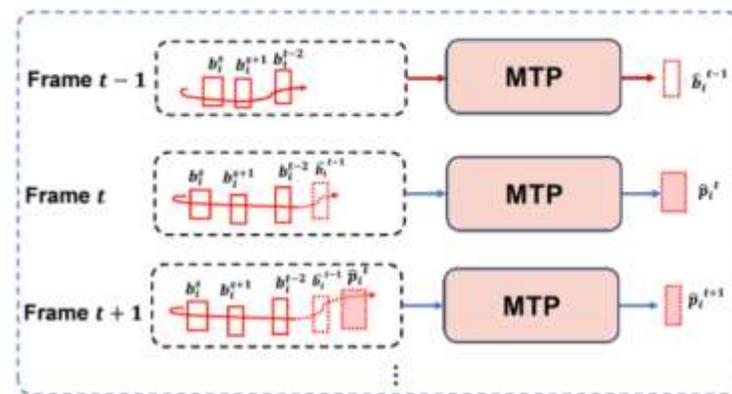


Figure 3: In TPM, we utilize MTP in an autoregressive manner to extend the lost tracklets, providing an opportunity for their trajectories to be re-established in future frames.

Detection
  Tracked object
  Predicted bounding box
  Patched bounding box
 ~ Active tracklet
 ~ Lost tracklet

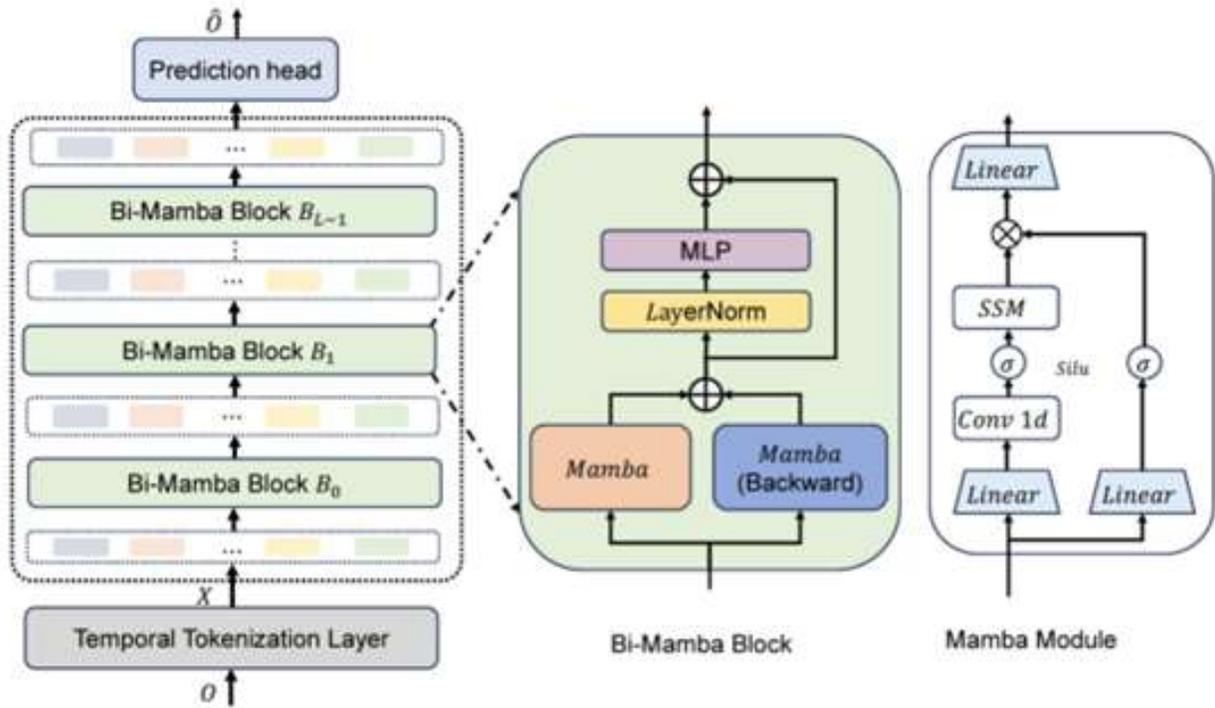


Figure 2: Overview of the proposed Mamba motion predictor.

**Temporal Tokenization Layer.** For a tracklet  $i$  in  $\mathbb{T}$ , we first construct the input trajectory feature:

$$\mathbf{O}_{in} = [\mathbf{o}_{t-q}, \mathbf{o}_{t-q+1}, \dots, \mathbf{o}_{t-1}] \in \mathbb{R}^{q \times 4}, \quad (4)$$

where  $q$  is the size of the look-back temporal window and  $\mathbf{o} = [\delta c_x, \delta c_y, \delta w, \delta h]$ , with  $\delta c_x$ ,  $\delta c_y$ ,  $\delta w$ , and  $\delta h$  representing the normalized changes of the corresponding bounding box center, width, and height between two observation time steps. We utilize a single linear layer to obtain the input token sequence as follows:

$$\mathbf{X} = \text{Embedding}(\mathbf{O}_{in}), \quad (5)$$

where  $\mathbf{X} \in \mathbb{R}^{q \times d_m}$  and  $d_m$  is the dimension of the temporal token.

**Bi-Mamba Encoding Layer.** After obtaining the temporal tokens  $\mathbf{X}$  of tracklets, we feed them into the designed bi-Mamba encoding layer to explore the motion patterns from the object's dynamic history. The bi-Mamba encoding layer comprises  $L$  bi-Mamba blocks. Specifically, to fully utilize the information from the object trajectory and address the unidirectional limitation of Mamba, each bi-Mamba block contains bidirectional Mamba modules: one forward and one backward. For the  $l$ -th bi-Mamba block, the inference process can be formulated as follows:

$$\begin{aligned} \hat{\mathbf{X}}_{forward} &= \text{Mamba}(\mathbf{X}_{l-1}), \\ \hat{\mathbf{X}}_{backward} &= \text{Mamba}_{backward}(\mathbf{X}_{l-1}), \\ \hat{\mathbf{Y}} &= \hat{\mathbf{X}}_{forward} + \hat{\mathbf{X}}_{backward}, \\ \mathbf{X}_l &= \hat{\mathbf{Y}} + \text{LN}(\text{MLP}(\hat{\mathbf{Y}})), \end{aligned} \quad (6)$$

where  $\mathbf{X}_{l-1}$  is the output of the  $(l-1)$ -th bi-Mamba block, LN is the layer normalization function [1], and MLP is a two-layer multi-layer perceptron. The selective SSM is the core of the Mamba [14] module which is described in Sec. 3.

**Prediction head and training.** After being processed by the bi-Mamba encoding layer, an average pooling layer is utilized to aggregate the information from  $\mathbf{X}_l$ . Subsequently, a prediction head comprising two fully connected layers is employed to predict the offsets  $\hat{\mathbf{O}}$ . We utilize the smooth L1 loss to supervise the training process:

$$L(\hat{\mathbf{O}}, \mathbf{O}^*) = \frac{1}{4} \sum \text{smooth}_{L_1}(\hat{\delta}_i - \delta_i), i \in \{c_x, c_y, w, h\}, \quad (7)$$

where  $\mathbf{O}^* = \{\delta c_x, \delta c_y, \delta w, \delta h\}$  represents the ground truth.

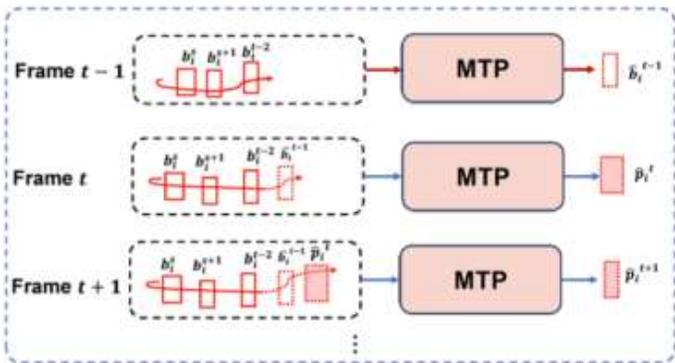


Figure 3: In TPM, we utilize MTP in an autoregressive manner to extend the lost tracklets, providing an opportunity for their trajectories to be re-established in future frames.

Table 4: Comparison of different motion models.

	HOTA↑	IDF1↑	AssA↑	MOTA↑	DetA↑
None(IoU)	44.7	36.8	25.3	87.3	<b>79.6</b>
KF	45.9	50.9	30.7	86.3	69.0
LSTM	51.3	51.6	34.4	87.1	76.7
TF	52.5	52.5	35.2	<b>89.3</b>	78.5
MTP	<b>54.9</b>	<b>54.5</b>	<b>38.5</b>	<b>89.3</b>	78.6

For example, if a lost tracklet  $\mathcal{T}_i$  in lost tracklets  $\mathbb{T}_{lost}$  receives no new update at the last time step  $t-1$  and remains unmatched at the current frame  $t$ , we compensate for this missing observation in an autoregressive manner by considering the predicted bounding box  $\hat{\mathbf{b}}_i^{t-1}$  as the actual observation of frame  $t-1$ . We then continue to predict its spatial location  $\hat{\mathbf{p}}_i^t$  at the current frame. As shown in Figure 3, if it still fails to match with a new detection in the current frame, we persist in predicting its future bounding boxes frame by frame utilizing the motion predictor MTP, leveraging the historical trajectory sequence  $\mathcal{T}_{past} = \{\dots, \mathbf{b}_i^s, \mathbf{b}_i^{s+1}, \dots, \hat{\mathbf{b}}_i^{t-1}\}$  and the predicted bounding box  $\hat{\mathbf{p}}_i^t$  in an autoregressive manner:

$$\hat{\mathbf{p}}_i^{t+1} = \text{MTP}(\mathcal{T}_{past}, \hat{\mathbf{p}}_i^t). \quad (8)$$

Since the bounding boxes obtained through autoregression for lost tracklets are typically less reliable compared to those of active tracklets, we prioritize the association of active tracklets with the detection results  $\hat{\mathcal{B}}_t$  in the current frame. Therefore, the active tracklets are given precedence in being associated with the detection results in the current frame. The remaining detection results are then associated with  $\hat{\mathcal{P}}_t$ , the predicted bounding boxes of the lost tracklets. The detailed inference process is described below.

Table 5: Apply MTP to other SORT-like trackers.

Tracker	w/ MTP	HOTA↑	DetA↑	AssA↑	MOTA↑
SORT[3]		45.9	50.9	30.7	86.3
	✓	<b>54.9 (+9.0)</b>	<b>54.5</b>	<b>38.5</b>	<b>89.3</b>
ByteTrack[54]		47.1	70.5	31.5	88.2
	✓	<b>53.9 (+6.8)</b>	<b>78.7</b>	<b>37.1</b>	<b>89.7</b>
MixSort[9]		46.7	53.0	31.9	85.8
	✓	<b>52.4 (+5.7)</b>	<b>76.7</b>	<b>36.0</b>	<b>87.3</b>

Algorithm 1: Inference of MambaTrack at frame  $t$ .

```

Input: Detections:  $\mathcal{B}_t = \{\mathbf{b}_i^t\}_{i=1}^M$ , tracklets  $\mathbb{T} = \{\mathcal{T}_j\}_{j=1}^N$  at frame
 $t-1$ , Motion Predictor: MTP.
Output: Active tracklets  $\mathbb{T}_{active}$  at current frame  $t$ .

/* First Matching */
1  $\mathbb{T}_{active}, \mathbb{T}_{lost} \leftarrow \mathbb{T}$ 
2  $\mathcal{B}_t \leftarrow [\mathbf{b}_1^t, \dots, \mathbf{b}_t^t]$  // Detection set of current frame
3  $\hat{\mathcal{B}}_t \leftarrow [\hat{\mathbf{b}}_1^t, \dots, \hat{\mathbf{b}}_t^t]$  from  $\mathbb{T}_{active}$  // Predicted bounding
boxes
4  $\mathcal{C}_t \leftarrow C_{IoU}(\hat{\mathcal{B}}_t, \mathcal{B}_t)$  // Cost matrix based on IoU similarity
5  $\mathcal{M}, \mathbb{T}_u, \mathcal{B}_u \leftarrow \text{Hungarian}(\mathcal{C}_t)$ 
6  $\mathbb{T}_{active} \leftarrow \{\mathcal{T}_i.update(\mathbf{b}_i^t), \forall (i, j) \in \mathcal{M}\}$ 
/* Re-find lost tracklets via patched bounding boxes. */
7  $\mathbb{T}_{lost} \leftarrow \mathbb{T}_{lost} \cup \mathbb{T}_u$  // Lost tracklets
8  $\hat{\mathcal{P}} \leftarrow [\hat{\mathbf{p}}_1^t, \dots, \hat{\mathbf{p}}_t^t]$  from  $\mathbb{T}_{lost}$ 
9  $\mathcal{C}_{lost} \leftarrow C_{IoU}(\hat{\mathcal{P}}, \mathcal{B}_u)$ 
10  $\mathcal{M}, \mathbb{T}_u, \mathcal{B}_u \leftarrow \text{Hungarian}(\mathcal{C}_{lost})$ 
/* Second Matching */
/* Add the re-find lost tracklets to active tracklets */
11  $\mathbb{T}_{active} \leftarrow \{\mathcal{T}_i.update(\hat{\mathbf{p}}_i^t), \forall (i, j) \in \mathcal{M}\}$ 
/* Update the lost tracklets with last predicted
bounding boxes */
12 for  $\mathcal{T}$  in  $\mathbb{T}_{lost}$  do
13 |  $\mathcal{T}.update(\mathcal{T}.\hat{\mathbf{b}}_{t-1}^t)$ 
14 end
15  $\mathbb{T} \leftarrow \mathbb{T}_{lost} \cup \mathbb{T}_{active}$ 
/* Predict next bounding boxes of tracklets */
16 for  $\mathcal{T}$  in  $\mathbb{T}$  do
17 |  $\text{MTP}(\mathcal{T})$ 
18 end
    
```

Table 7: Impact of different numbers  $L$  of Bi-Mamba blocks.

$L$	HOTA↑	IDF1↑	AssA↑	MOTA↑	DetA↑
1	52.1	51.8	34.7	89.2	78.5
2	54.1	54.4	37.4	<b>89.3</b>	<b>78.7</b>
3	<b>54.9</b>	<b>54.5</b>	<b>38.5</b>	<b>89.3</b>	78.6
4	52.1	52.1	34.7	<b>89.3</b>	78.5
5	52.3	52.4	35.1	<b>89.3</b>	78.3



# 感谢倾听