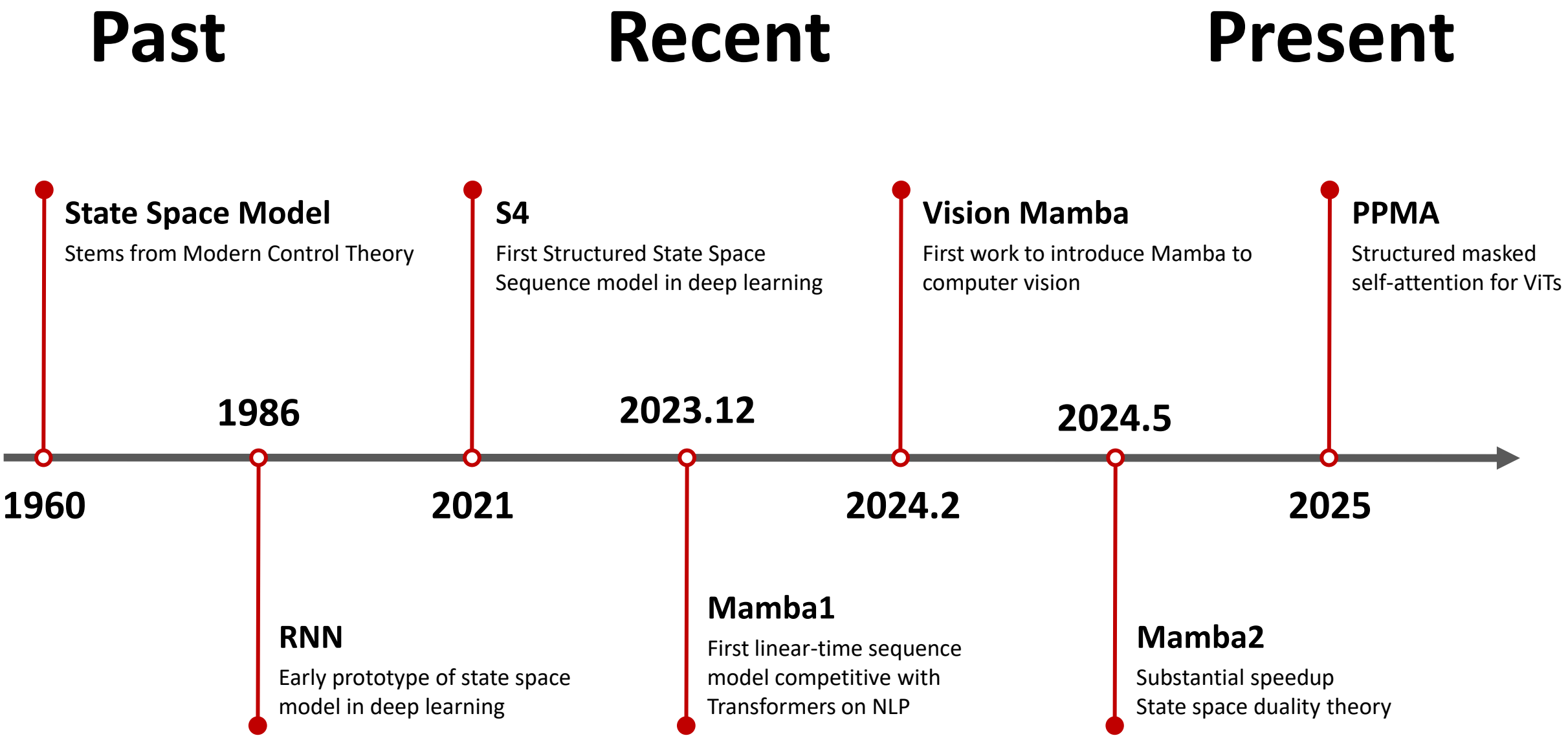
西安交通大学
XI'AN JIAOTONG UNIVERSITY

From Mamba to Structured Masked Attention: Past, Recent, and Present

Zhongchen Zhao

2025/09/22


Past
Recent
Present
State Space Model
Stems from Modern Control Theory
1960
1986
RNN
Early prototype of state space model in deep learning
S4
First Structured State Space Sequence model in deep learning
2021
2023.12
Mamba1
First linear-time sequence model competitive with Transformers on NLP
Vision Mamba
First work to introduce Mamba to computer vision
2024.2
2024.5
Mamba2
Substantial speedup
State space duality theory
PPMA
Structured masked self-attention for ViTs
2025

目录

1. State Space Model: 现代控制理论 (1960s)

- 状态空间方程（源于**现代控制理论**）
  - 考虑一个**质量-弹簧-阻尼系统**的受力分析，一个质量为m的物体块质点
    - 系统输入：外力 $x(t)$
    - 系统输出：质量块的位置 $y(t)$
    - 微分方程 (牛顿第二定律: $F = ma$)

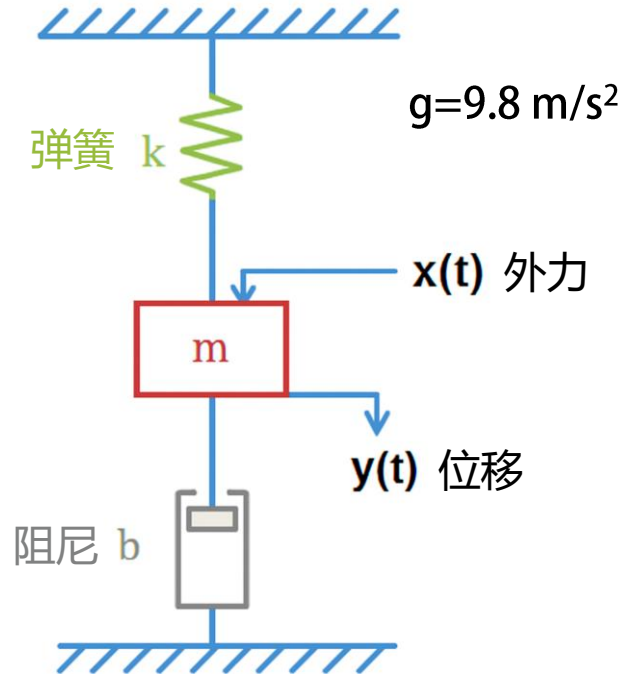
$$m \times y''(t) = x(t) + mg - k \times y(t) - b \times y'(t)$$

    - 状态空间方程：使用**一阶微分方程组**就可以表示**任意阶线性系统**

$$\begin{bmatrix} y'(t) \\ y''(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} (x(t) + mg)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$$

状态向量　输入变量　输出变量



1. State Space Model: 现代控制理论 (1960s)

- 状态空间方程（源于现代控制理论）
  - 考虑一个**质量-弹簧-阻尼系统**的受力分析，一个质量为m的物体块质点
    - 状态空间方程：使用**一阶微分方程组**就可以表示**任意阶线性系统**

参数 A　　参数 B

$$\begin{bmatrix} y'(t) \\ y''(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} (x(t) + mg)$$

状态向量　　输入变量

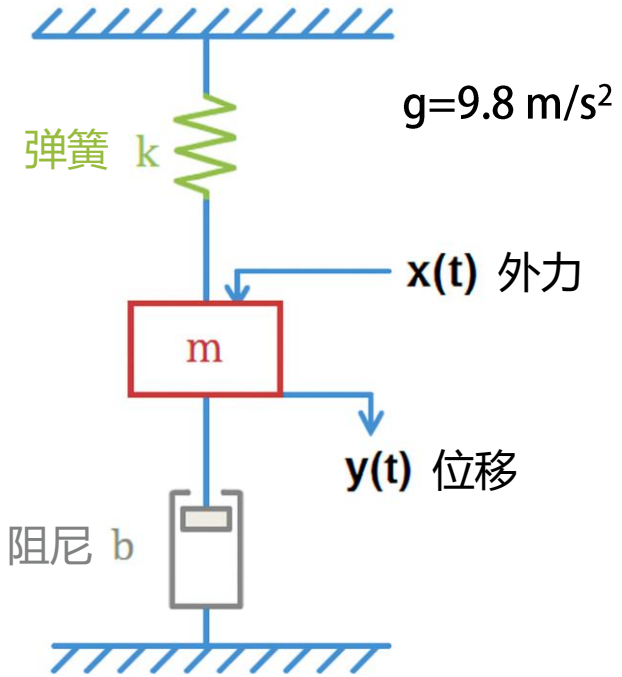
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$$

输出变量　　参数 C

隐变量

- 记 $\boldsymbol{h}(t) = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$，

$$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A} \times \boldsymbol{h}(t) + \boldsymbol{B} \times x(t) \\ y(t) = \boldsymbol{C} \times \boldsymbol{h}(t) \end{cases}$$



1. State Space Model: 现代控制理论 (1960s)

- 状态空间方程（源于现代控制理论）
  - 用向量和矩阵形式描述**动态系统**的方法，能够统一表示单输入单输出（**SISO**）和多输入多输出（**MIMO**）系统，适用于**连续**或**离散**、**线性**或**非线性**、**时变**或**时不变**系统
  - 对于**连续时间**、**线性时不变**系统，状态空间方程为：

$$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A} \times \boldsymbol{h}(t) + \boldsymbol{B} \times x(t) \\ y(t) = \boldsymbol{C} \times \boldsymbol{h}(t) \end{cases}$$

» Structured State Space Sequence Model (S4)

  - 对于**连续时间**、**线性时变**系统，状态空间方程为：

$$\begin{cases} \boldsymbol{h}'(t) = \boxed{\boldsymbol{A}(t)} \times \boldsymbol{h}(t) + \boxed{\boldsymbol{B}(t)} \times x(t) \\ y(t) = \boxed{\boldsymbol{C}(t)} \times \boldsymbol{h}(t) \end{cases}$$

» Selective State Space Model (Mamba)

1. State Space Model: 现代控制理论 (1960s)

- 状态空间方程（源于现代控制理论）
  - 用向量和矩阵形式描述**动态系统**的方法，能够统一表示单输入单输出（**SISO**）和多输入多输出（**MIMO**）系统，适用于**连续**或**离散**、**线性**或**非线性**、**时变**或**时不变**系统

系统类型	状态空间方程
连续线性时不变	$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A} \times \boldsymbol{h}(t) + \boldsymbol{B} \times x(t) \\ y(t) = \boldsymbol{C} \times \boldsymbol{h}(t) \end{cases}$
连续线性时变	$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A}(t) \times \boldsymbol{h}(t) + \boldsymbol{B}(t) \times x(t) \\ y(t) = \boldsymbol{C}(t) \times \boldsymbol{h}(t) \end{cases}$
连续非线性时不变	$\begin{cases} \boldsymbol{h}'(t) = f(\boldsymbol{h}(t), x(t)) \\ y(t) = g(\boldsymbol{h}(t), x(t)) \end{cases}$
连续非线性时变	$\begin{cases} \boldsymbol{h}'(t) = f(\boldsymbol{h}(t), x(t), t) \\ y(t) = g(\boldsymbol{h}(t), x(t), t) \end{cases}$

系统类型	状态空间方程
离散线性时不变 S4	$\begin{cases} \boldsymbol{h}[k+1] = \overline{\boldsymbol{A}} \times \boldsymbol{h}[k] + \overline{\boldsymbol{B}} \times x[k] \\ y[k] = \overline{\boldsymbol{C}} \times \boldsymbol{h}[k] \end{cases}$
离散线性时变 Mamba	$\begin{cases} \boldsymbol{h}[k+1] = \overline{\boldsymbol{A}}[k] \times \boldsymbol{h}[k] + \overline{\boldsymbol{B}}[k] \times x[k] \\ y[k] = \overline{\boldsymbol{C}}[k] \times \boldsymbol{h}[k] \end{cases}$
离散非线性时不变 RNN	$\begin{cases} \boldsymbol{h}[k+1] = f(\boldsymbol{h}[k], x[k]) \\ y[k] = g(\boldsymbol{h}[k], x[k]) \end{cases}$
离散非线性时变	$\begin{cases} \boldsymbol{h}[k+1] = f(\boldsymbol{h}[k], x[k], k) \\ y[k] = g(\boldsymbol{h}[k], x[k], k) \end{cases}$

1. State Space Model: 现代控制理论 (1960s)

- 连续状态空间方程的**离散化**
  - e^x的多种近似方法：
    - a) $\lim_{x\to 0} e^x \approx 1+x$ ✓
    - b) $\lim_{x\to 0} e^x \approx \frac{1+x/2}{1-x/2}$

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

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

$$\Rightarrow \boldsymbol{h}'(t)e^{-\boldsymbol{A}t} - \boldsymbol{A}\boldsymbol{h}(t)e^{-\boldsymbol{A}t} = \boldsymbol{B}x(t)\, e^{-\boldsymbol{A}t}$$

$$\Rightarrow \int \boldsymbol{h}(t)e^{-\boldsymbol{A}t}dt = \int \boldsymbol{B}x(t)\, e^{-\boldsymbol{A}t}dt$$

Δ: 时间采样间隔

设 $\boldsymbol{g}(t) = \boldsymbol{h}(t)e^{-\boldsymbol{A}t} \quad \Rightarrow \boldsymbol{g}'(t) = e^{-\boldsymbol{A}t}\boldsymbol{B}x(t)$

$$\therefore \boldsymbol{h}(t+\Delta)e^{-\boldsymbol{A}t-\Delta\boldsymbol{A}} - \boldsymbol{h}(t)e^{-\boldsymbol{A}t} = \boldsymbol{g}(t+\Delta) - \boldsymbol{g}(t)$$

$$= \int_t^{t+\Delta} \boldsymbol{g}'(t)dt$$

$$= \int_t^{t+\Delta} e^{-\boldsymbol{A}t}\boldsymbol{B}x(t)\, dt$$

$$\approx \int_t^{t+\Delta} e^{-\boldsymbol{A}t}\boldsymbol{B}\, dt \cdot x(t) \quad (if\ \Delta\to 0)$$

$$= \boldsymbol{A}^{-1}e^{-\boldsymbol{A}t}\left(I - e^{-\Delta\boldsymbol{A}}\right)\boldsymbol{B}\cdot x(t)$$

$$\Rightarrow \boldsymbol{h}(t+\Delta) = e^{\Delta\boldsymbol{A}}\boldsymbol{h}(t) + \boldsymbol{A}^{-1}(e^{\Delta\boldsymbol{A}} - I)\boldsymbol{B}x(t)$$

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

where $\overline{\boldsymbol{A}} = \boldsymbol{e}^{\Delta\boldsymbol{A}}$, $\overline{\boldsymbol{B}} = \boldsymbol{A}^{-1}(\boldsymbol{e}^{\Delta\boldsymbol{A}} - \boldsymbol{I})\boldsymbol{B} \approx \Delta\boldsymbol{B}$

1. State Space Model: Structured State Space Model

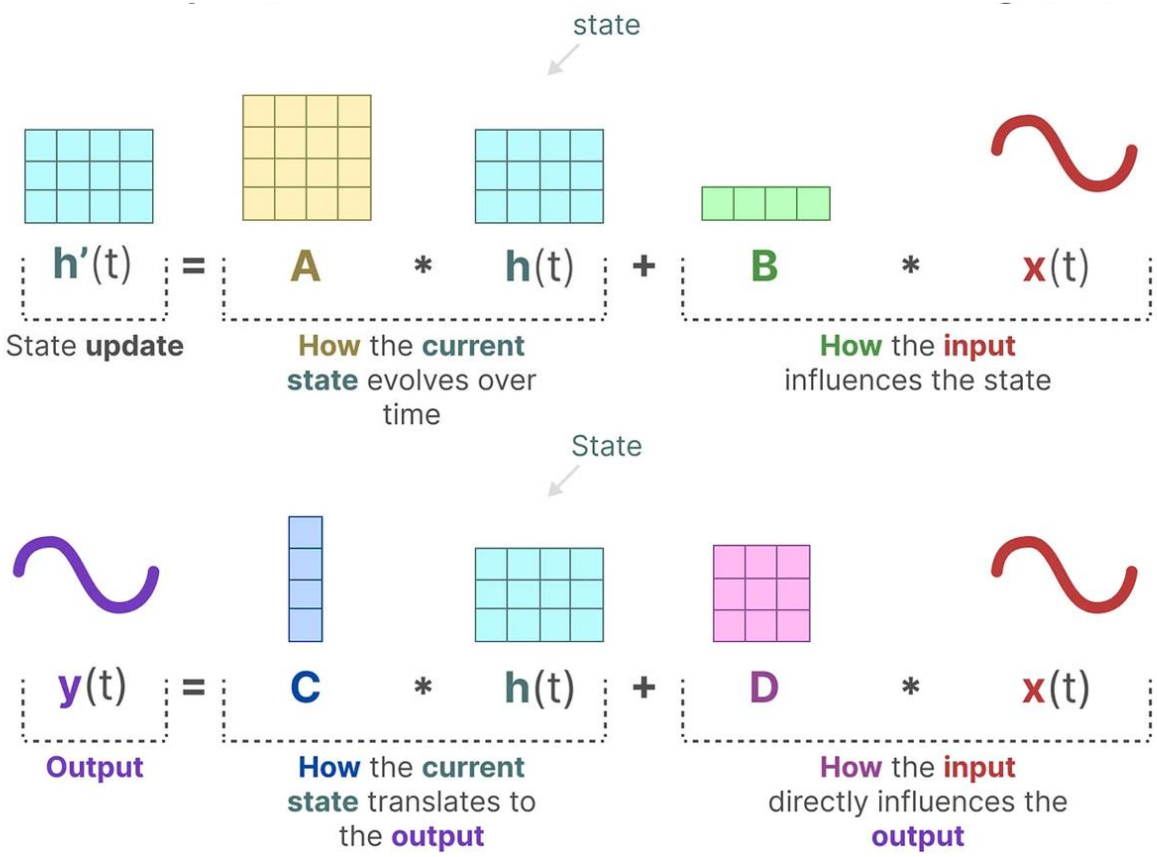
- Structured State Space Model (**S4**)
  - **S4**将系统建模为**线性时不变**模型，旨在弥补 Transformer/RNN 在长序列建模中的不足
    - **连续**时间的状态空间方程

$$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A} \times \boldsymbol{h}(t) + \boldsymbol{B} \times x(t) \\ y(t) = \boldsymbol{C} \times \boldsymbol{h}(t) + D \times x(t) \end{cases}$$

    - **离散**时间的状态空间方程

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + D \times x_t \end{cases}$$

    - 变量 $x_t, y_t \in \mathbb{R}$, $\boldsymbol{h}_t \in \mathbb{R}^{\mathrm{D}}$
    - 参数 $\overline{\boldsymbol{A}} = \boldsymbol{e}^{\Delta \boldsymbol{A}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$, $\overline{\boldsymbol{B}} = \Delta \boldsymbol{B} \in \mathbb{R}^{\mathrm{D}\times 1}$
    - 参数 $\boldsymbol{C} \in \mathbb{R}^{1\times\mathrm{D}}$, $D \in \mathbb{R}$



[1] **Gu A**, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces[C]. ICLR, **2022**. (Cited: 2309)

1. State Space Model: Structured State Space Model

- Structured State Space Model (S4)
  - **S4**将系统建模为**线性时不变**模型，旨在弥补 Transformer/RNN 在长序列建模中的不足
    - **连续**时间的状态空间方程

$$\begin{cases} \boldsymbol{h}'(t) = \boldsymbol{A} \times \boldsymbol{h}(t) + \boldsymbol{B} \times x(t) \\ y(t) = \boldsymbol{C} \times \boldsymbol{h}(t) + D \times x(t) \end{cases}$$

    - **离散**时间的状态空间方程

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + D \times x_t \end{cases}$$

    - 变量 $x_t, y_t \in \mathbb{R}$，　$\boldsymbol{h}_t \in \mathbb{R}^{\mathrm{D}}$
    - 参数 $\overline{\boldsymbol{A}} = \boldsymbol{e}^{\Delta \boldsymbol{A}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$，　$\overline{\boldsymbol{B}} = \Delta \boldsymbol{B} \in \mathbb{R}^{\mathrm{D}\times 1}$
    - 参数 $\boldsymbol{C} \in \mathbb{R}^{1\times\mathrm{D}}$，　$D \in \mathbb{R}$

- 参数 $\overline{\boldsymbol{A}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$ 是一个HiPPO矩阵，定义为：

$$\overline{\boldsymbol{A}}_{i,j} = \begin{cases} \sqrt{(2i+1)(2j+1)} & if\ i > j\ and\ i+j\ is\ odd \\ 0 & otherwise \end{cases}$$

- 例如，$D = 5$

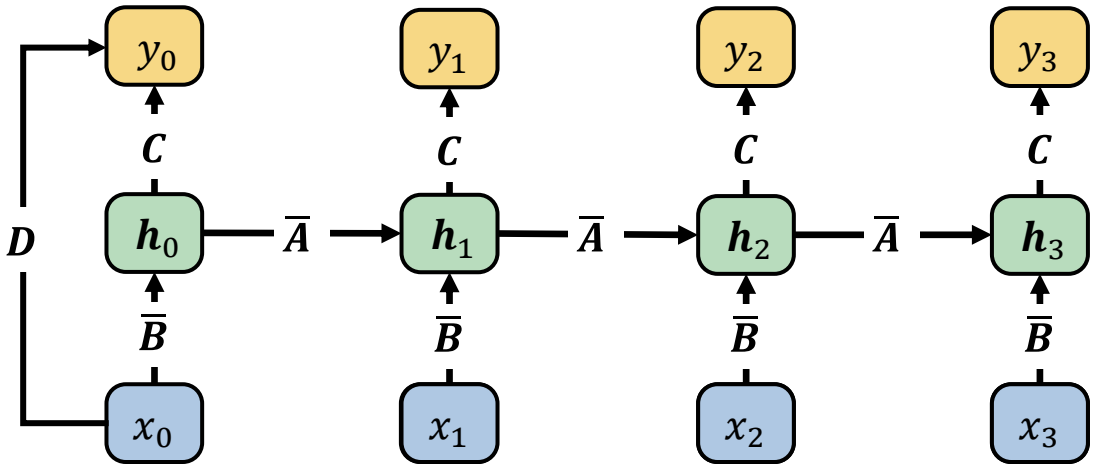
$$\overline{\boldsymbol{A}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \sqrt{3} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{15} & 0 & 0 & 0 \\ \sqrt{7} & 0 & \sqrt{35} & 0 & 0 \\ 0 & \sqrt{27} & 0 & \sqrt{63} & 0 \end{bmatrix}$$

[1] **Gu A**, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces[C]. ICLR, **2022**. (Cited: 2309)

1. Structured State Space Model: Parallelizable Training

- Structured State Space Model (S4)
  - 状态空间模型的**循环递推**形式（**串行计算，速度慢**）

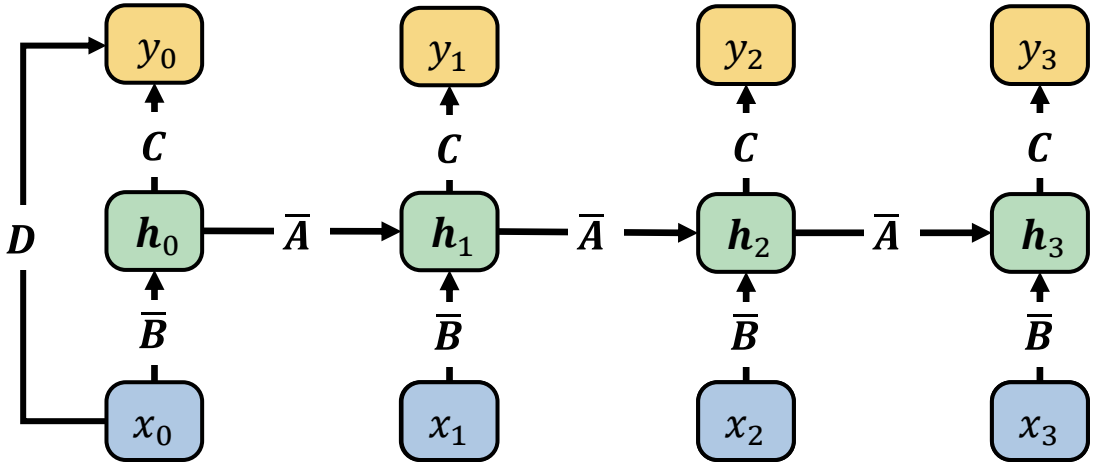
$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + \cancel{D \times x_t} \end{cases}$$



1. Structured State Space Model: Parallelizable Training

- Structured State Space Model (S4)
  - 状态空间模型的**循环递推**形式（**串行计算，速度慢**）

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + \cancel{D \times x_t} \end{cases}$$



  - 状态空间模型的**一维卷积**形式（**并行计算，速度快**）

$$y_0 = \boldsymbol{C}\,\overline{\boldsymbol{B}} x_0$$

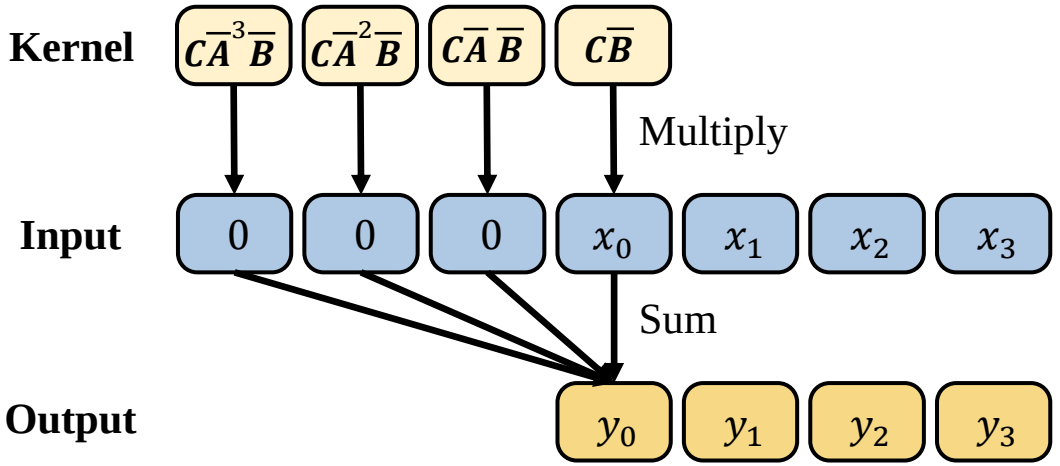
$$y_1 = \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_1$$

$$y_2 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_2$$

$$\cdots$$

$$y_t = \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}^{t-1}\overline{\boldsymbol{B}} x_1 + \cdots + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_{t-1} + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_t$$

$$y_t = \overline{\boldsymbol{K}} * \vec{\boldsymbol{x}} = \sum_{n=0}^{t} \overline{\boldsymbol{K}}_n x_{t-n}, \quad \overline{\boldsymbol{K}} = [\boldsymbol{C}\,\overline{\boldsymbol{B}}, \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}}, \ldots, \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}}]$$

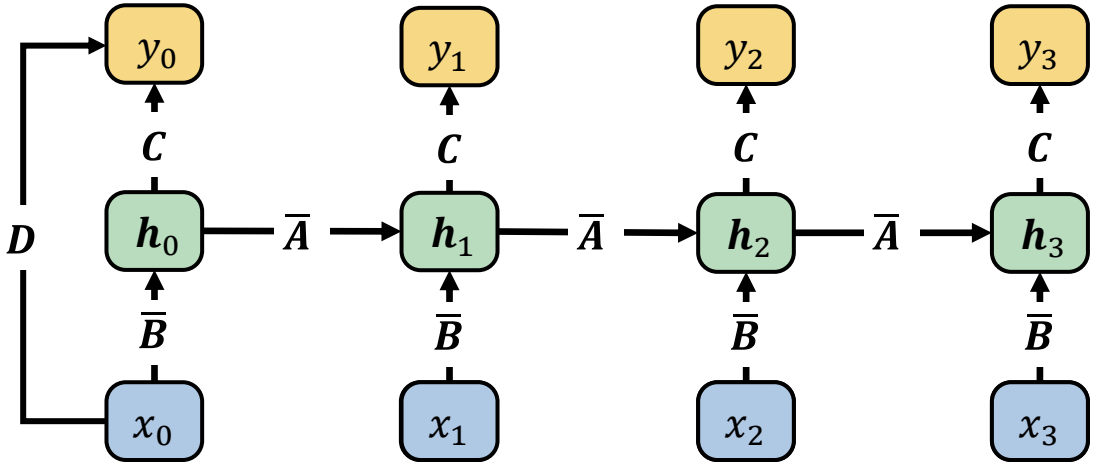


$y_0 = \boldsymbol{C}\,\overline{\boldsymbol{B}} x_0$

1. Structured State Space Model: Parallelizable Training

- Structured State Space Model (S4)
  - 状态空间模型的**循环递推**形式（**串行计算，速度慢**）

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + \cancel{D \times x_t} \end{cases}$$



  - 状态空间模型的**一维卷积**形式（**并行计算，速度快**）

$$y_0 = \boldsymbol{C}\,\overline{\boldsymbol{B}} x_0$$

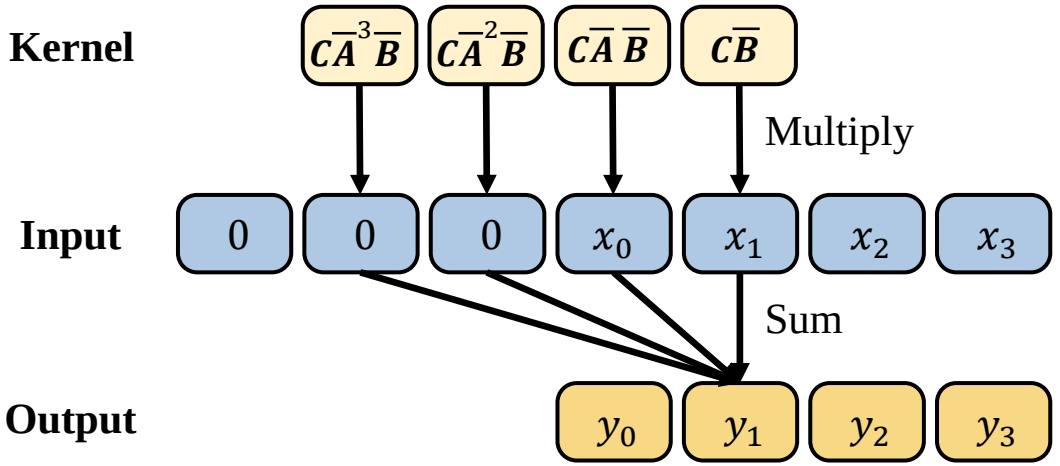
$$y_1 = \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_1$$

$$y_2 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_2$$

$$\cdots$$

$$y_t = \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}^{t-1}\overline{\boldsymbol{B}} x_1 + \cdots + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_{t-1} + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_t$$

$$y_t = \overline{\boldsymbol{K}} * \vec{\boldsymbol{x}} = \sum_{n=0}^{t} \overline{\boldsymbol{K}}_n x_{t-n}, \quad \overline{\boldsymbol{K}} = [\boldsymbol{C}\,\overline{\boldsymbol{B}}, \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}}, \ldots, \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}}]$$

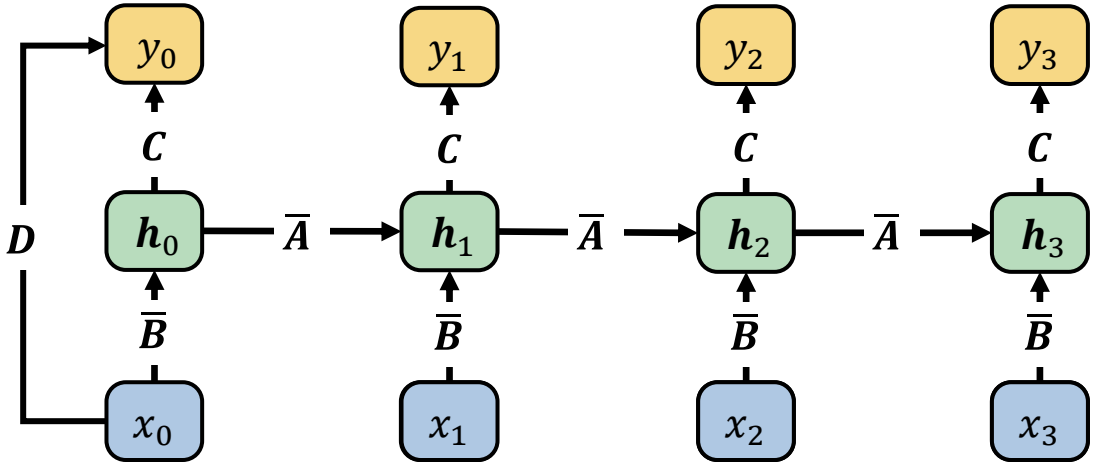


$$y_1 = \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_1$$

1. Structured State Space Model: Parallelizable Training

- Structured State Space Model (S4)
  - 状态空间模型的**循环递推**形式（**串行计算，速度慢**）

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + \cancel{D \times x_t} \end{cases}$$



  - 状态空间模型的**一维卷积**形式（**并行计算，速度快**）

$$y_0 = \boldsymbol{C}\,\overline{\boldsymbol{B}} x_0$$

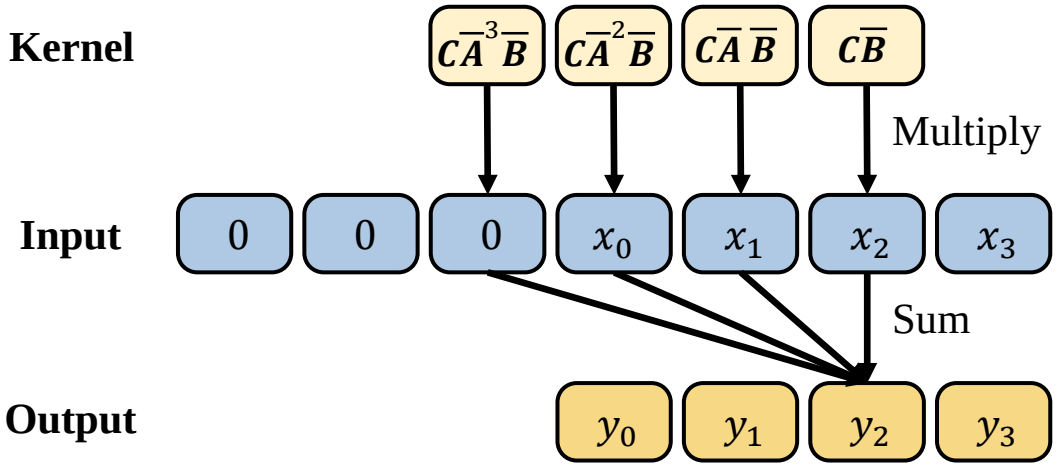
$$y_1 = \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_1$$

$$y_2 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_2$$

$$\cdots$$

$$y_t = \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}^{t-1}\overline{\boldsymbol{B}} x_1 + \cdots + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_{t-1} + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_t$$

$$y_t = \overline{\boldsymbol{K}} * \vec{\boldsymbol{x}} = \sum_{n=0}^{t} \overline{\boldsymbol{K}}_n x_{t-n}, \quad \overline{\boldsymbol{K}} = [\boldsymbol{C}\,\overline{\boldsymbol{B}}, \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}}, \ldots, \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}}]$$

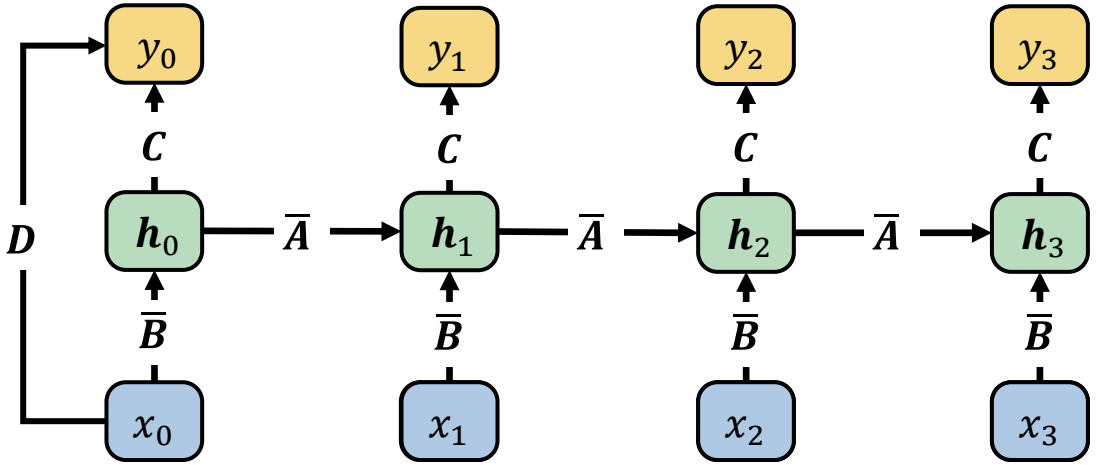


$$y_2 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_2$$

1. Structured State Space Model: Parallelizable Training

- Structured State Space Model (S4)
  - 状态空间模型的**循环递推**形式（**串行计算，速度慢**）

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t + \cancel{D \times x_t} \end{cases}$$



  - 状态空间模型的**一维卷积**形式（**并行计算，速度快**）

$$y_0 = \boldsymbol{C}\,\overline{\boldsymbol{B}} x_0$$

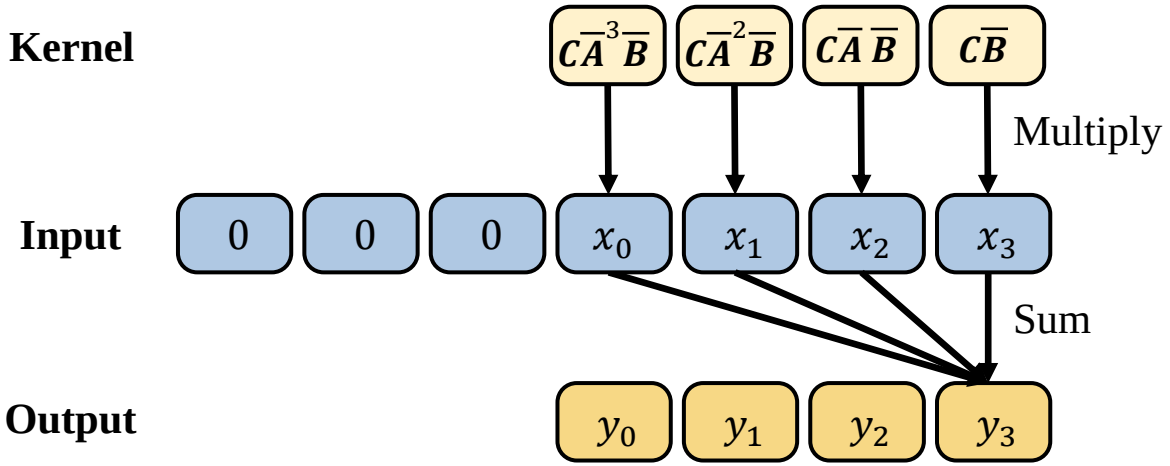
$$y_1 = \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_1$$

$$y_2 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_2$$

$$\cdots$$

$$y_t = \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}^{t-1}\overline{\boldsymbol{B}} x_1 + \cdots + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_{t-1} + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_t$$

$$y_t = \overline{\boldsymbol{K}} * \vec{\boldsymbol{x}} = \sum_{n=0}^{t} \overline{\boldsymbol{K}}_n x_{t-n}, \quad \overline{\boldsymbol{K}} = [\boldsymbol{C}\,\overline{\boldsymbol{B}}, \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}}, \ldots, \boldsymbol{C}\,\overline{\boldsymbol{A}}^t\overline{\boldsymbol{B}}]$$



$$y_3 = \boldsymbol{C}\,\overline{\boldsymbol{A}}^3\,\overline{\boldsymbol{B}} x_0 + \boldsymbol{C}\,\overline{\boldsymbol{A}}^2\,\overline{\boldsymbol{B}} x_1 + \boldsymbol{C}\,\overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}} x_2 + \boldsymbol{C}\,\overline{\boldsymbol{B}} x_3$$

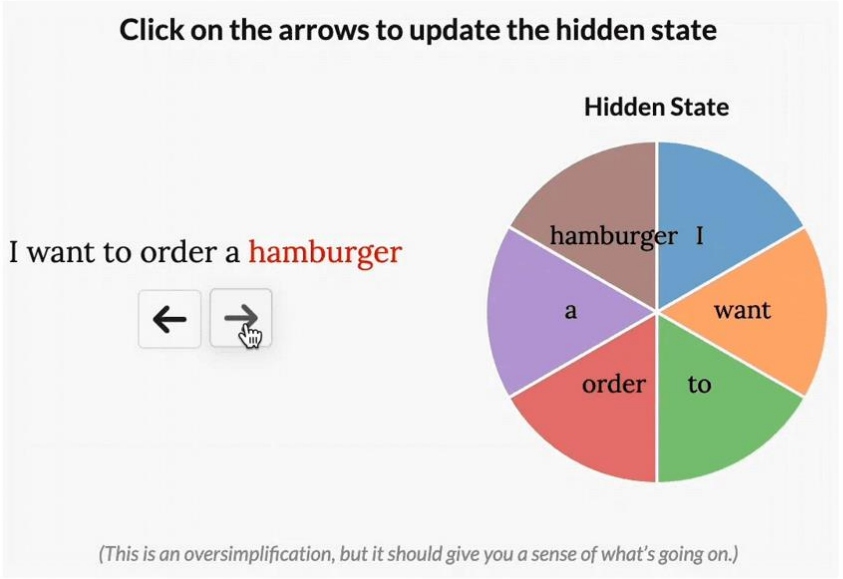
西安交通大学
XI'AN JIAOTONG UNIVERSITY

目录

2. Mamba1: Selectivity

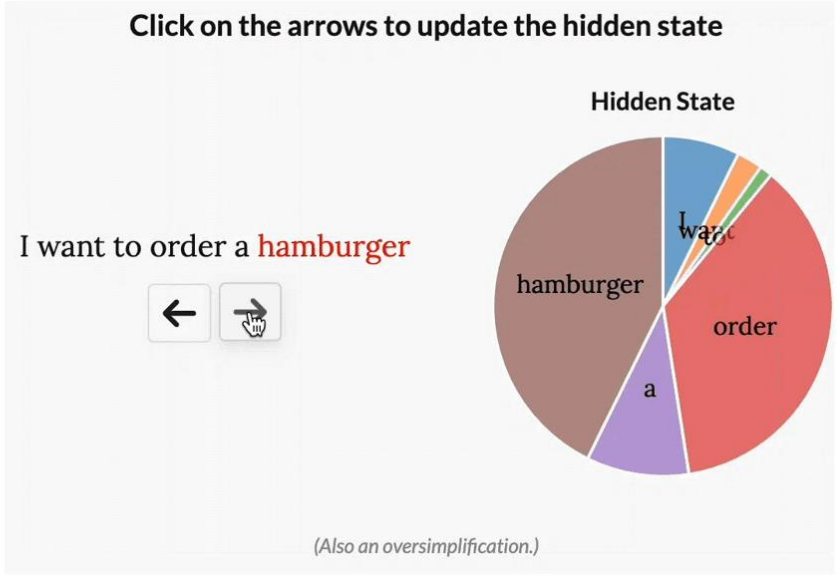
- Structured State Space Model (S4)

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}} \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}} \times x_t \\ y_t = \boldsymbol{C} \times \boldsymbol{h}_t \end{cases}$$



- **Selective** State Space Models (Mamba)

$$\begin{cases} \boldsymbol{h}_t = \boxed{\overline{\boldsymbol{A}}_t} \times \boldsymbol{h}_{t-1} + \boxed{\overline{\boldsymbol{B}}_t} \times x_t \\ y_t = \boxed{\boldsymbol{C}_t} \times \boldsymbol{h}_t \end{cases}$$



Selectivity：动态权重，对不同的输入赋以不同的权重

[1] **Gu A**, **Dao T**. Mamba: Linear-time sequence modeling with selective state spaces [C]. arXiv, **2023/12/01**. (cited: 3961)

2. Mamba1: Selectivity

- **Selective** State Space Models (Mamba)

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}}_t \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}_t \times x_t \\ y_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

网络的改变

$$B = nn.\text{Parameter}() \rightarrow B_t = nn.Linear(x_t)$$

$$C = nn.\text{Parameter}() \rightarrow C_t = nn.Linear(x_t)$$

$$\Delta = nn.\text{Parameter}() \rightarrow \Delta_t = nn.Linear(x_t)$$

Algorithm 1 SSM (S4)

Input: x : (B, L, D)
Output: y : (B, L, D)
1: $\boldsymbol{A}$: (D, N) ← Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B}$: (D, N) ← Parameter
3: $\boldsymbol{C}$: (D, N) ← Parameter
4: Δ : (D) ← τ_Δ(Parameter)
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}$: (D, N) ← 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 : (B, L, D)
Output: y : (B, L, D)
1: $\boldsymbol{A}$: (D, N) ← Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B}$: (B, L, N) ← $s_B(x)$
3: $\boldsymbol{C}$: (B, L, N) ← $s_C(x)$
4: Δ : (B, L, D) ← τ_Δ(Parameter+$s_\Delta(x)$)
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}$: (B, L, D, N) ← 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

2. Mamba1: Selectivity

- **Selective** State Space Models (Mamba)
  - Input tokens $\boldsymbol{x} = [\boldsymbol{x}_0 \quad \boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_{\mathrm{N}-1}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$, output tokens $\boldsymbol{y} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$ and hidden states $\boldsymbol{h} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}\times\mathrm{D}}$

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}}_t \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

  - Learnable input-dependent parameters

$$\boldsymbol{A} = -\exp\left(\mathrm{nn.Paramter}(torch.diag(\mathrm{D}))\right) \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}} \ (<0)$$

$$\boldsymbol{B}_t = \mathrm{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t \boldsymbol{W}_B \in \mathbb{R}^{1\times\mathrm{D}}$$

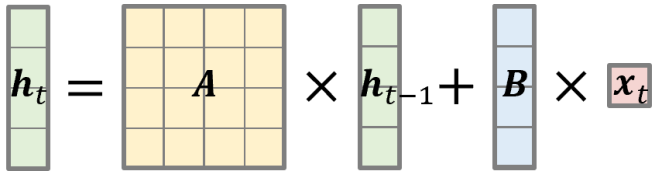
$$\boldsymbol{C}_t = \mathrm{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t \boldsymbol{W}_C \in \mathbb{R}^{1\times\mathrm{D}}$$

$$\Delta_t = \mathrm{Softplus}(\mathrm{MLP}_\Delta(\boldsymbol{x}_t)) \in \mathbb{R}^1 \ (>0)$$

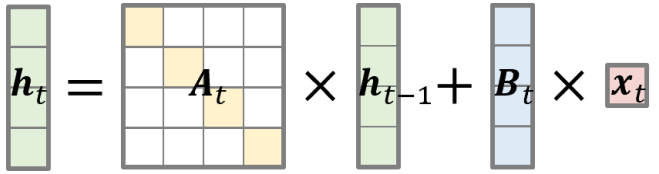
$$\overline{\boldsymbol{A}}_t = e^{\Delta_t \boldsymbol{A}} = \exp(\Delta_t \boldsymbol{A}) \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}} \ \mathbf{(0\sim1)}$$

$$\overline{\boldsymbol{B}}_t = \Delta_t \boldsymbol{B}_t \in \mathbb{R}^{1\times\mathrm{D}}$$

降低多个矩阵乘法 $\overline{\boldsymbol{A}}_t \cdots \overline{\boldsymbol{A}}_1$ 的计算量



(a) General (Unstructured) SSMs



(b) Mamba1 (Structured SSMs)

2. Mamba1: Parallelizable Training

- **深度学习模型的计算效率**
  - **训练推理速度 Throughput (images/s)** $\propto \frac{\text{并行度} \times \text{GPU算力}}{\text{计算复杂度}}$

训练推理时间/速度	基本概念	相关因素
计算复杂度 / 浮点运算次数 (FLOPs)	一次前向传播所需的总计算次数 (浮点数乘法加法的总次数)	模型的结构、输入数据的尺寸
并行度	可并发计算的单元数	模型的结构、进程/线程数、硬件架构（CUDA 核心数）

  - **存储空间**

存储空间	基本概念	相关因素
显存/内存占用	模型在运行过程中占用的内存资源	模型参数、前向激活值、反向传播的梯度等
模型存储占用	模型储存所占用的内存资源	模型参数量、参数的数据精度类型（FP32/FP16）

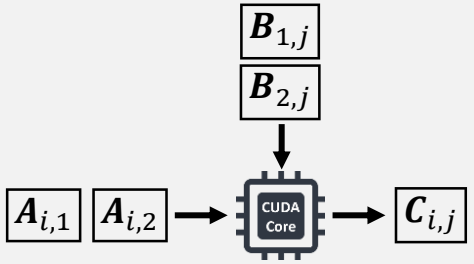
2. Mamba1: Parallelizable Training

- **深度学习模型的计算效率**
  - **训练推理速度 Throughput (images/s)：前向计算时间** $\propto \frac{\text{计算复杂度}}{\text{并行度} \times \text{GPU算力}}$

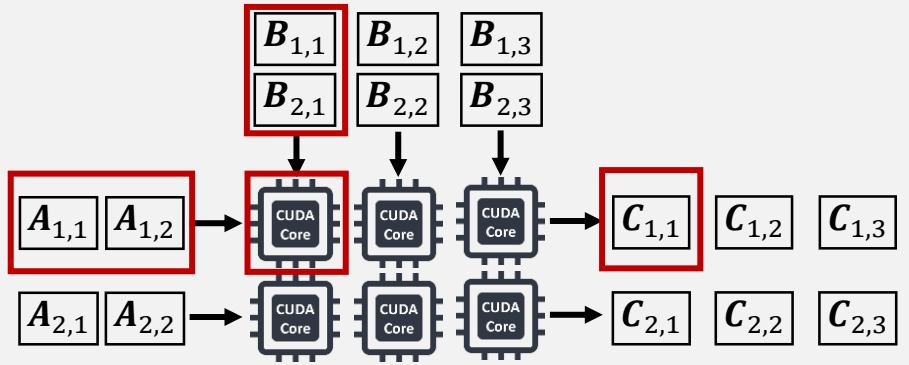
训练推理时间/速度	基本概念	相关因素
计算复杂度 / 浮点运算次数 (FLOPs)	一次前向传播所需的总计算次数 (浮点数乘法加法的总次数)	模型的结构、输入数据的尺寸
并行度	可并发计算的单元数	模型的结构、进程/线程数、硬件架构（CUDA 核心数）

  - 例如，矩阵乘法可利用的**并行计算**极大加速：$\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$，$\boldsymbol{A} \in \mathbb{R}^{2\times2}$，$\boldsymbol{B} \in \mathbb{R}^{2\times3}$ (FLOPs=12)

单进程 (总耗时 = $6 \times t_{\boldsymbol{C}_{i,j}}$)



多进程 (总耗时 = $t_{\boldsymbol{C}_{i,j}}$)



2. Mamba1: Parallelizable Training

- **Selective** State Space Models (Mamba)
  - Mamba1 的**循环递推**形式（**串行计算，速度慢**）

$$\begin{cases} \boldsymbol{h}_t = \overline{\boldsymbol{A}}_t \times \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

  - Mamba1 的**累计和**形式（**并行计算，速度快**）

$$\boldsymbol{y}_0 = \boldsymbol{C}_0\, \overline{\boldsymbol{B}}_0 \boldsymbol{x}_0$$

$$\boldsymbol{y}_1 = \boldsymbol{C}_1\, \overline{\boldsymbol{A}}_1\, \overline{\boldsymbol{B}}_0 \boldsymbol{x}_0 + \boldsymbol{C}_1 \overline{\boldsymbol{B}}_1 \boldsymbol{x}_1$$

$$\boldsymbol{y}_2 = \boldsymbol{C}_2 \overline{\boldsymbol{A}}_2 \overline{\boldsymbol{A}}_1 \overline{\boldsymbol{B}}_0 \boldsymbol{x}_0 + \boldsymbol{C}_2 \overline{\boldsymbol{A}}_2 \overline{\boldsymbol{B}}_1 \boldsymbol{x}_1 + \boldsymbol{C}_2 \overline{\boldsymbol{B}}_2 \boldsymbol{x}_2$$

$$\dots$$

$$\boldsymbol{y}_t = \boldsymbol{C}_t \left(\overline{\boldsymbol{A}}_t \cdots \overline{\boldsymbol{A}}_1\right) \overline{\boldsymbol{B}}_0 \boldsymbol{x}_0 + \boldsymbol{C}_t \left(\overline{\boldsymbol{A}}_t \cdots \overline{\boldsymbol{A}}_2\right) \overline{\boldsymbol{B}}_1 \boldsymbol{x}_1 + \cdots + \boldsymbol{C}_t \overline{\boldsymbol{A}}_t\, \overline{\boldsymbol{B}}_{t-1} \boldsymbol{x}_{t-1} + \boldsymbol{C}_t \overline{\boldsymbol{B}}_t \boldsymbol{x}_t$$

$$y_0 = C_0 \overline{B}_0 x_0$$

$$y_1 = C_1 \overline{A}_1 \overline{B}_0 x_0 + C_1 \overline{B}_1 x_1$$

$$y_2 = C_2 \overline{A}_2 \overline{A}_1 \overline{B}_0 x_0 + C_2 \overline{A}_2 \overline{B}_1 x_1 + C_2 \overline{B}_2 x_2$$

$$\cdots$$

$$y_t = C_t \left(\overline{A}_t \cdots \overline{A}_1\right) \overline{B}_0 x_0 + C_t \left(\overline{A}_{t-1} \cdots \overline{A}_1\right) \overline{B}_1 x_1 + \cdots + C_t \overline{A}_{t-1} \overline{B}_{t-1} x_{t-1} + C_t \overline{B}_t x_t$$

令 $\overline{x}_t = B_t x_t,\ y_t = C_t \overline{y}_t$

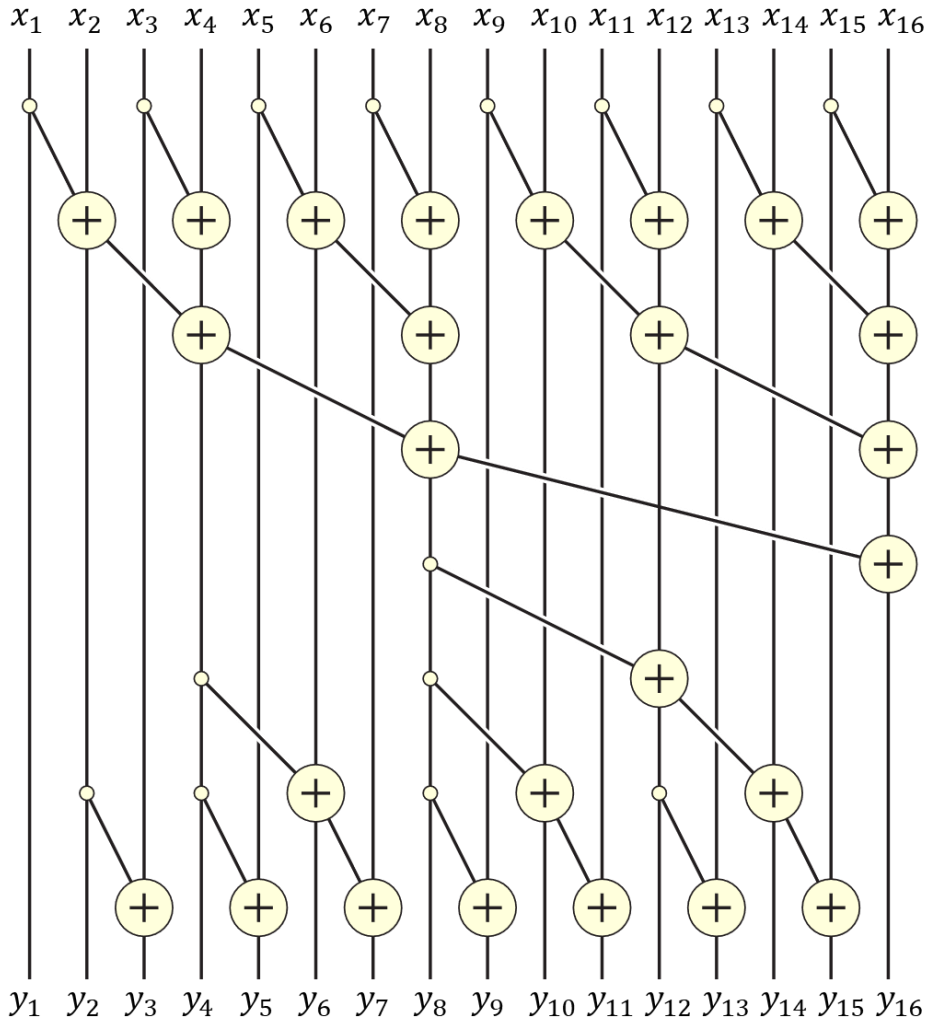
$$\overline{y}_0 = \overline{x}_0$$

$$\overline{y}_1 = \overline{A}_1 \overline{x}_0 + \overline{x}_1$$

$$\overline{y}_2 = \overline{A}_2 \overline{A}_1 \overline{x}_0 + \overline{A}_2 \overline{x}_1 + \overline{x}_2$$

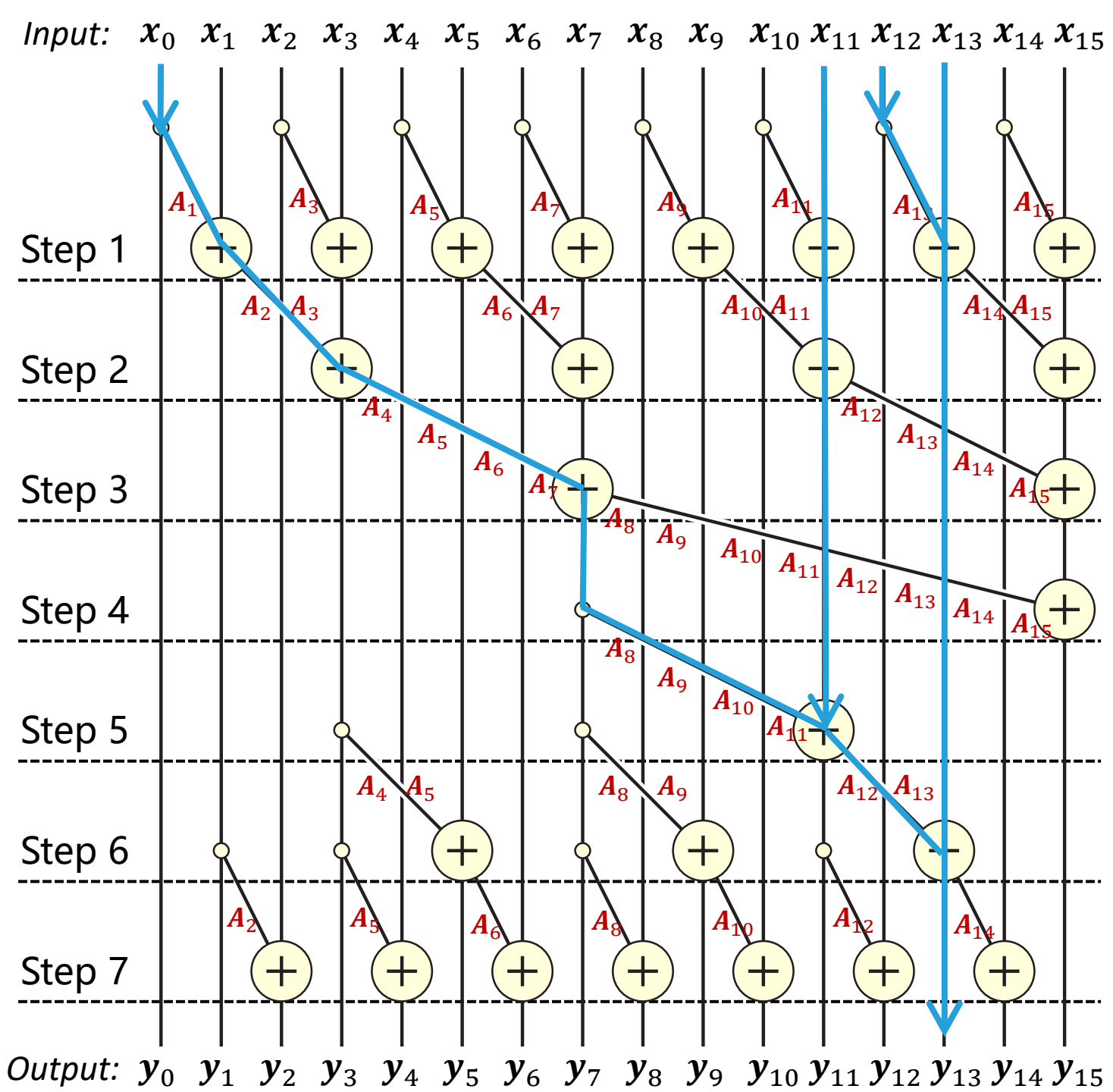
$$\overline{y}_3 = \overline{A}_3 \overline{A}_2 \overline{A}_1 \overline{x}_0 + \overline{A}_3 \overline{A}_2 \overline{x}_1 + \overline{A}_3 \overline{x}_2 + \overline{x}_3$$

$$\cdots$$



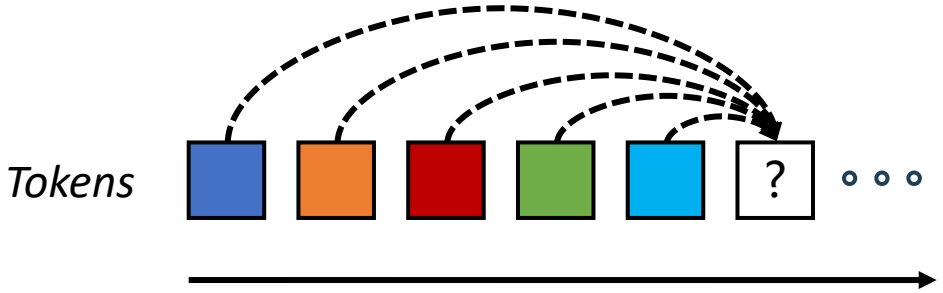
$$y_0 = x_0$$
$$y_1 = A_1 x_0 + x_1$$
$$y_2 = A_2 A_1 x_0 + A_2 x_1 + x_2$$
$$y_3 = A_3 A_2 A_1 x_0 + A_3 A_2 x_1 + A_3 x_2 + x_3$$
$$\dots$$

- 累计和 (Cumulative Sum)
  - 串行计算（从左向右求和）
    - 时间复杂度：$\mathcal{O}(N)$
  - 并行计算（从上向下求和）
    - 时间复杂度：$\mathcal{O}(\log N)$

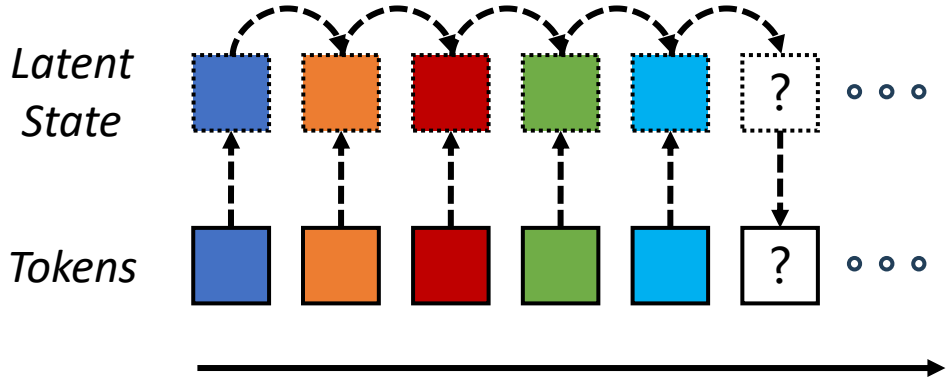


2. Mamba1

- **Selective** State Space Models (Mamba)



(a) Transformer (complexity: $\mathcal{O}(N^2)$)



(b) State Space Model (complexity: $\mathcal{O}(N)$)

Recurrent(RNN)

✓ Global receptive fields
Stateful inference

✗ Inefficient training
Vanishing gradients

Convolution (CNN)

✓ Easy optimization
Parallelizable training

✗ Bounded receptive fields

Transformer

✓ Global receptive fields
Parallelizable training

✗ Quadratic complexity

SSM (Mamba)

✓ **Global receptive fields**
Dynamic weights
Linear computational
✓ **Parallelizable training**

2. Mamba1: SOTA performance on NLP tasks

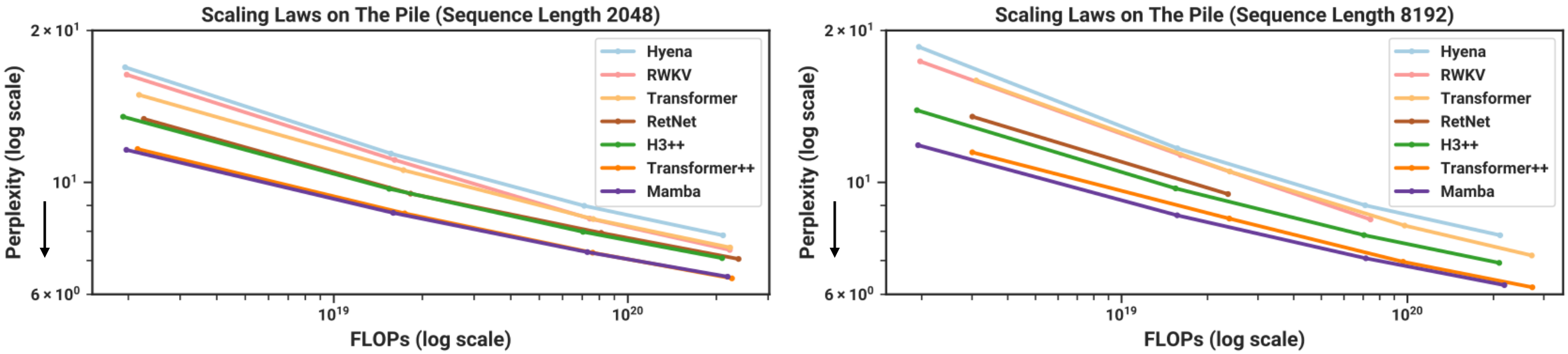


Figure 4: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence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

2. Mamba1: SOTA performance on NLP tasks

Table 3: (**Zero-shot Evaluations**.) Best results for each size in bold. We compare against open source LMs with various tokenizers, trained for up to 300B tokens. Pile refers to the validation split, comparing only against models trained on the same dataset and tokenizer (GPT-NeoX-20B). For each model size, Mamba is best-in-class on every single evaluation result, and generally matches baselines at twice the model size.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
Hybrid H3-130M	GPT2	—	89.48	25.77	31.7	64.2	44.4	24.2	50.6	40.1
Pythia-160M	NeoX	29.64	38.10	33.0	30.2	61.4	43.2	24.1	**51.9**	40.6
Mamba-130M	NeoX	**10.56**	**16.07**	**44.3**	**35.3**	**64.5**	**48.0**	**24.3**	**51.9**	**44.7**
Hybrid H3-360M	GPT2	—	12.58	48.0	41.5	68.1	51.4	24.7	54.1	48.0
Pythia-410M	NeoX	9.95	10.84	51.4	40.6	66.9	52.1	24.6	53.8	48.2
Mamba-370M	NeoX	**8.28**	**8.14**	**55.6**	**46.5**	**69.5**	**55.1**	**28.0**	**55.3**	**50.0**
Pythia-1B	NeoX	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	51.9
Mamba-790M	NeoX	**7.33**	**6.02**	**62.7**	**55.1**	**72.1**	**61.2**	**29.5**	**56.1**	**57.1**
GPT-Neo 1.3B	GPT2	—	7.50	57.2	48.9	71.1	56.2	25.9	54.9	52.4
Hybrid H3-1.3B	GPT2	—	11.25	49.6	52.6	71.3	59.2	28.1	56.9	53.0
OPT-1.3B	OPT	—	6.64	58.0	53.7	72.4	56.7	29.6	59.5	55.0
Pythia-1.4B	NeoX	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
RWKV-1.5B	NeoX	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	54.3
Mamba-1.4B	NeoX	**6.80**	**5.04**	**64.9**	**59.1**	**74.2**	**65.5**	**32.8**	**61.5**	**59.7**
GPT-Neo 2.7B	GPT2	—	5.63	62.2	55.8	72.1	61.1	30.2	57.6	56.5
Hybrid H3-2.7B	GPT2	—	7.92	55.7	59.7	73.3	65.6	32.3	61.4	58.0
OPT-2.7B	OPT	—	5.12	63.6	60.6	74.8	60.8	31.3	61.0	58.7
Pythia-2.8B	NeoX	6.73	5.04	64.7	59.3	74.0	64.1	32.9	59.7	59.1
RWKV-3B	NeoX	7.00	5.24	63.9	59.6	73.7	67.8	33.1	59.6	59.6
Mamba-2.8B	NeoX	**6.22**	**4.23**	**69.2**	**66.1**	**75.2**	**69.7**	**36.3**	**63.5**	**63.3**
GPT-J-6B	GPT2	–	4.10	68.3	66.3	75.4	67.0	36.6	64.1	63.0
OPT-6.7B	OPT	–	4.25	67.7	67.2	76.3	65.6	34.9	65.5	62.9
Pythia-6.9B	NeoX	6.51	4.45	67.1	64.0	75.2	67.3	35.5	61.3	61.7
RWKV-7.4B	NeoX	6.31	4.38	67.2	65.5	76.1	67.8	37.5	61.0	62.5

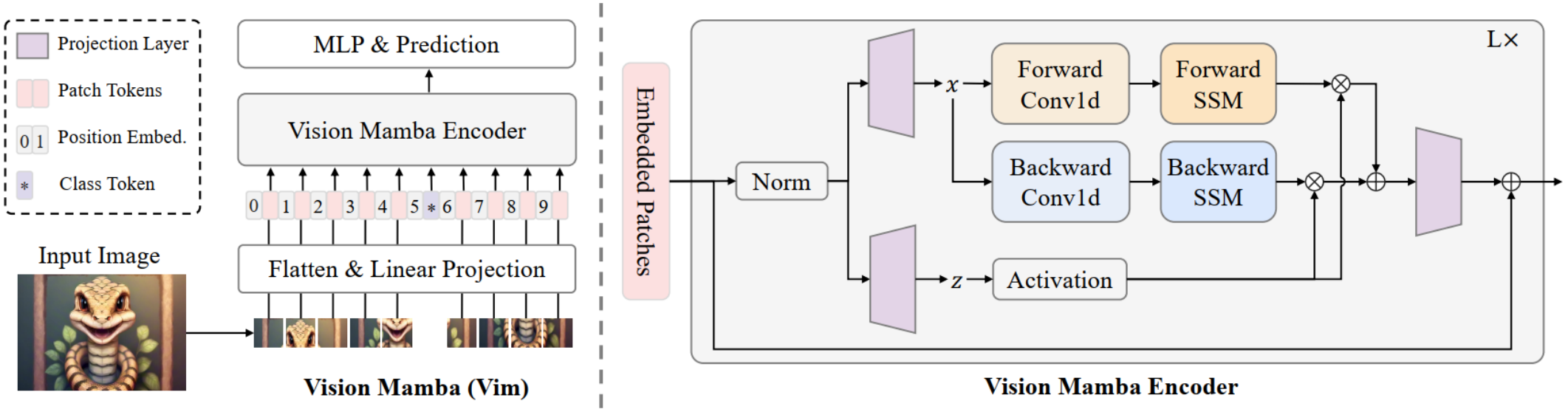
西安交通大学
XI'AN JIAOTONG UNIVERSITY

目录

3. Vision Mamba

- ViM: 如何将Mamba应用于视觉任务
  - Input tokens: $\boldsymbol{x} \in \mathbb{R}^{N \times C}$; output tokens: $\boldsymbol{y} \in \mathbb{R}^{N \times C}$
  - 类似于 ViTs, 将图像裁剪为 patches，展开成像文本一样的一维序列 输入Mamba block
  - 单向扫描 → 双向扫描

First work to introduce
Mamba to computer vision



[1] Xinggang Wang, et al. Vision mamba: Efficient visual representation learning with bidirectional state space model[C].**ICML**, **2024/01/17** (cited: 1726)

3. Vision Mamba

- VMamba
  - Cross scanning: four scanning paths in different directions

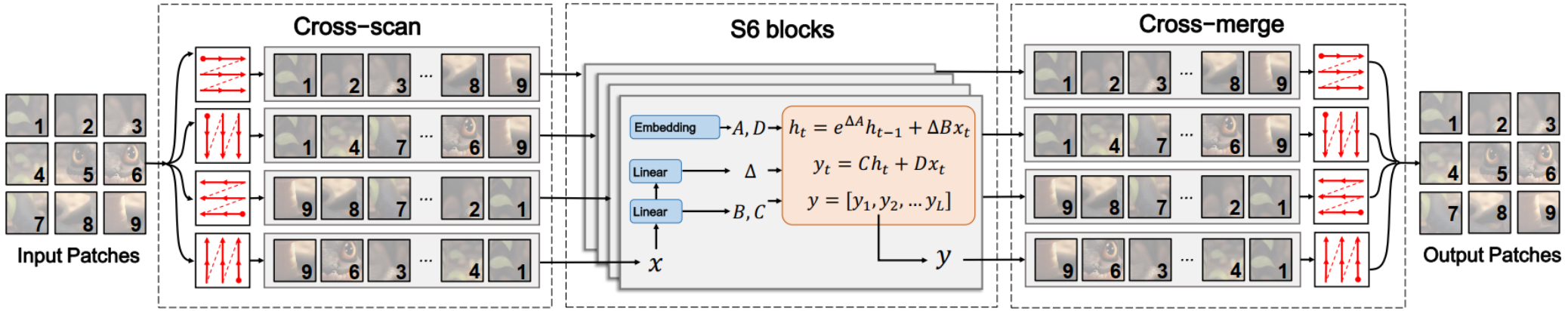
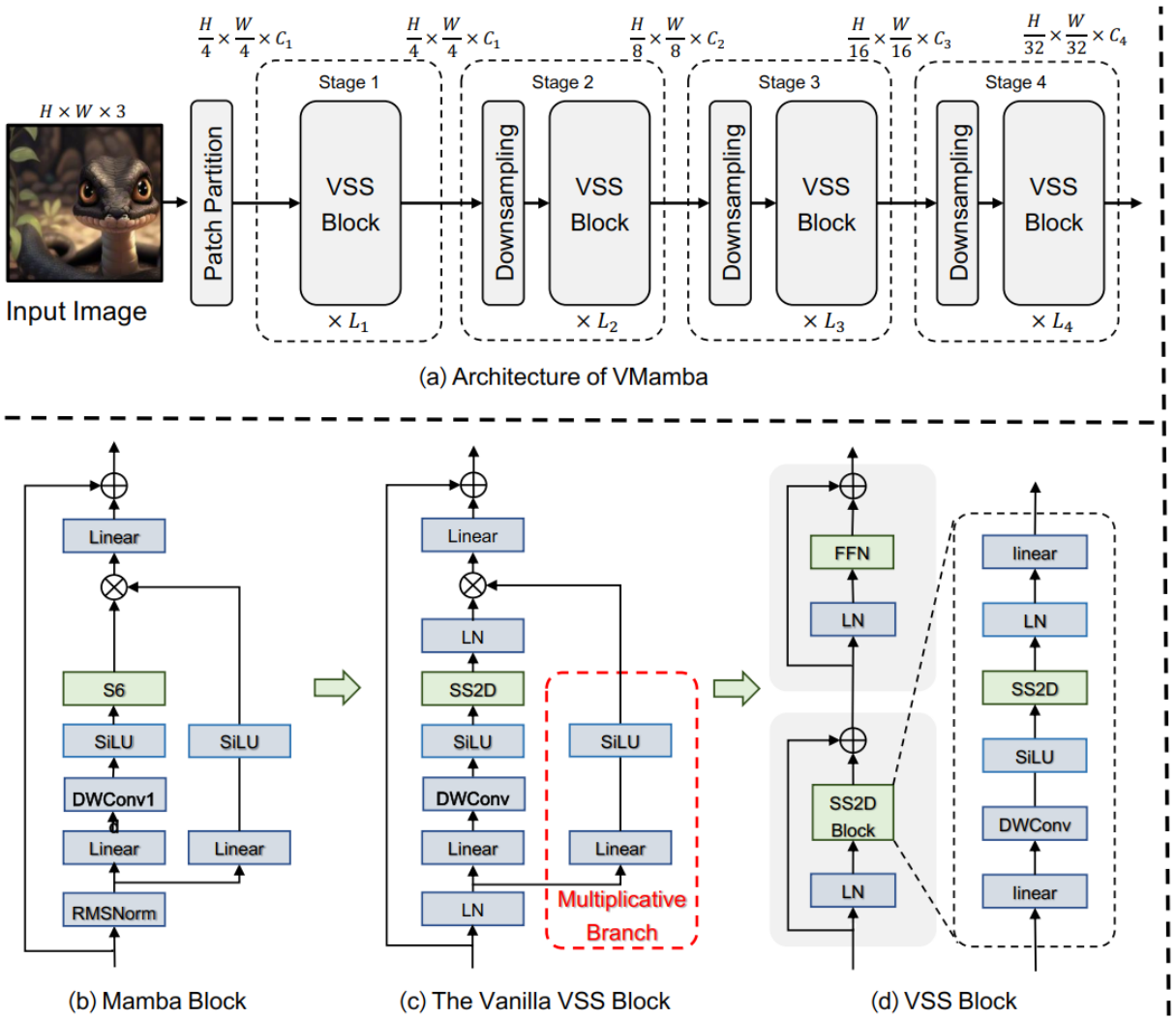


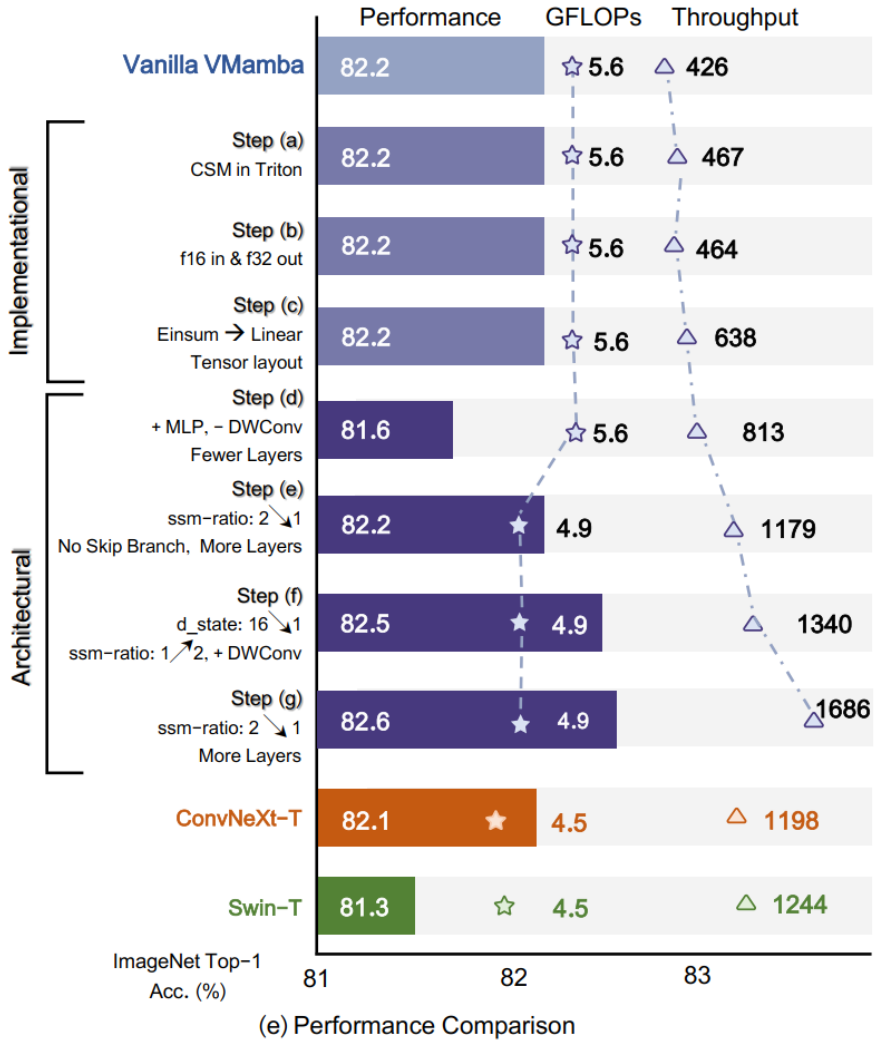
Figure 2: Illustration of 2D-Selective-Scan (SS2D). Input patches are traversed along four different scanning paths (*Cross-Scan*), and each sequence is independently processed by distinct S6 blocks. Subsequently, the results are merged to construct a 2D feature map as the final output (*Cross-Merge*).

[1] Yunfan Liu, et al. Vmamba: Visual state space model[C]. 2024 **NeurIPS**, **2024/01/18** (cited: 1733)

3. Vision Mamba

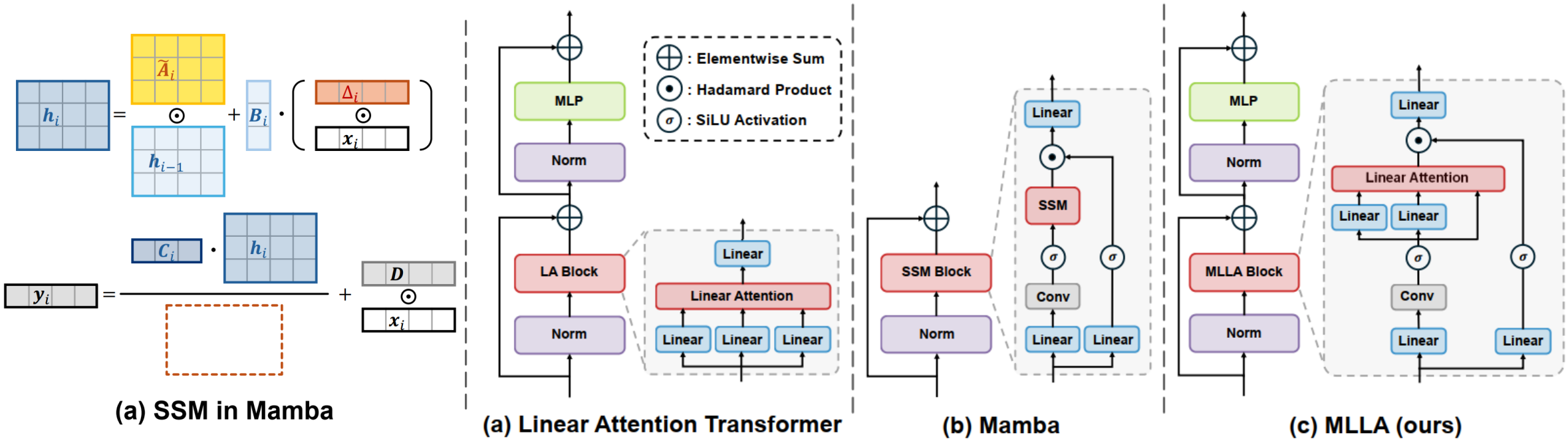
- VMamba





3. Vision Mamba

- MLLA: Mamba-Like Linear Attention
  - Reveal that Mamba model shares surprising similarities with linear attention Transformer



[1] Dongchen Han, Gao Huang, et al. Demystify mamba in vision: A linear attention perspective[C]. 2024 **NeurIPS**, **2024/05/26** (cited: 84)

3. Vision Mamba

- Mamba Vision
  - First Mamba-Transformer hybrid architecture for computer vision applications

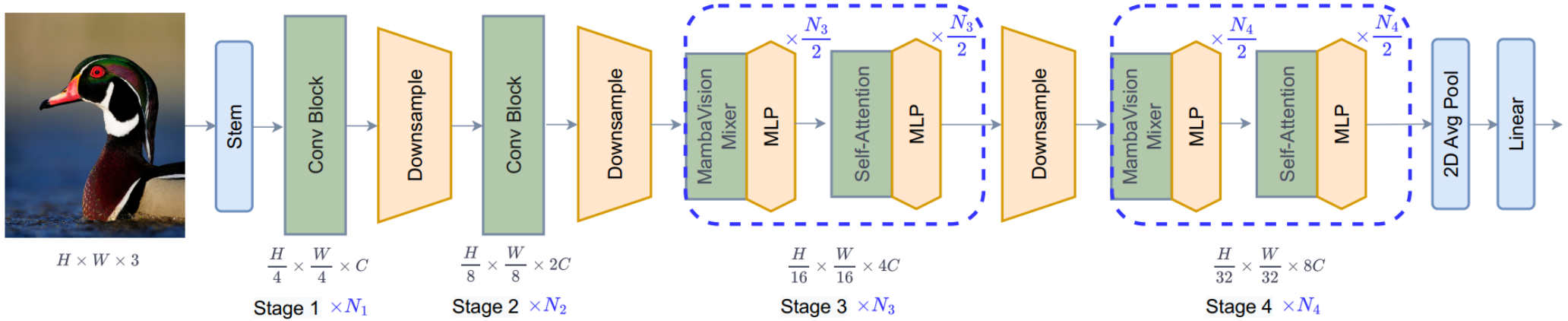


Figure 2 – The architecture of hierarchical MambaVision models. The first two stages use residual convolutional blocks for fast feature extraction. Stage 3 and 4 employ both MambaVision and Transformer blocks. Specifically, given N layers, we use $\frac{N}{2}$ MambaVision and MLP blocks which are followed by additional $\frac{N}{2}$ Transformer and MLP blocks. The Transformer blocks in final layers allow for recovering lost global context and capture long-range spatial dependencies.

[1] NVIDIA. Mambavision: A hybrid mamba-transformer vision backbone[C]. 2025 **CVPR**, **2024/07/10** (cited: 116)

3. Vision Mamba

- MambaIR
  - First to work to adapt state space models for low-level image restoration

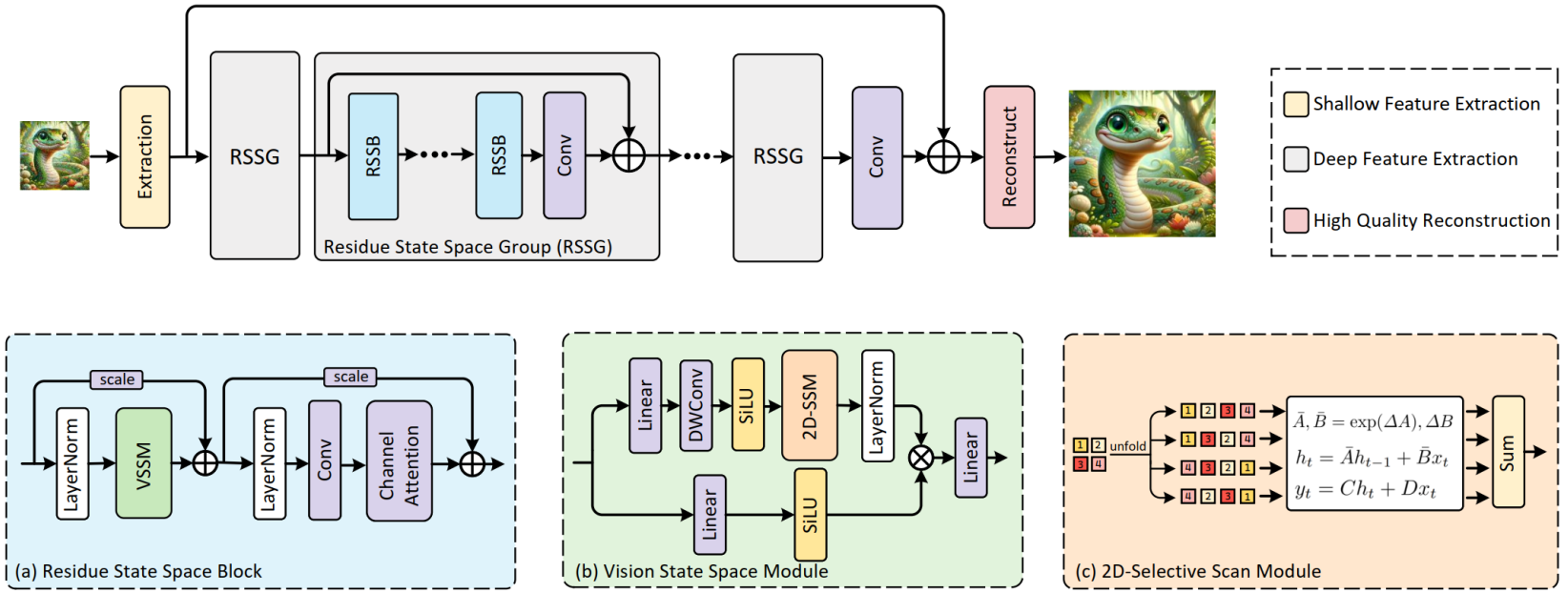


Fig. 2: The overall network architecture of our MambaIR, as well as the (a) Residual State-Space Block (RSSB), the (b) Vision State-Space Module (VSSM), and the (c) 2D Selective Scan Module (2D-SSM).

[1] Hang Guo, Jinmin Li, Tao Dai, et al. Mambair: A simple baseline for image restoration with state-space model[C]. 2024 ECCV, **2024/02/23** (cited: 382)

3. Vision Mamba

- MambaIR V2

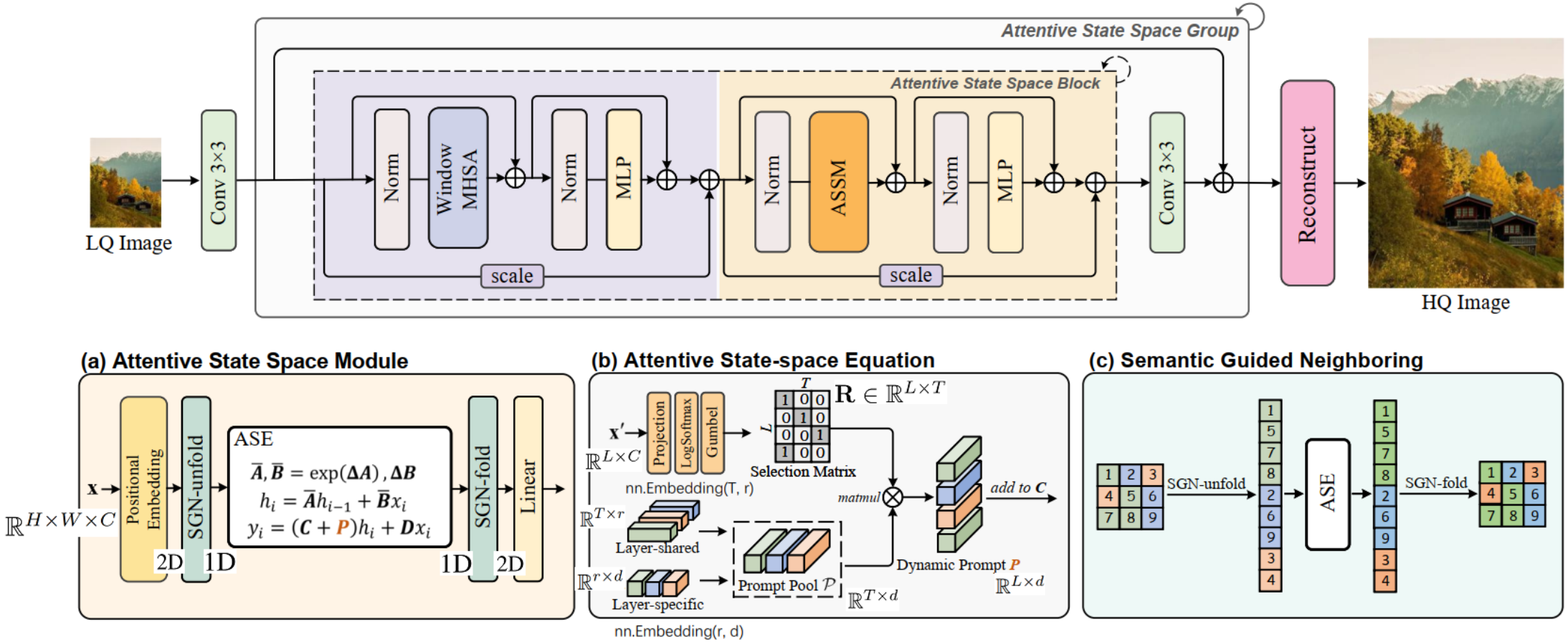


Figure 3. The overall architecture of our proposed MambaIRv2, as well as the (a) Attentive State Space Module (ASSM), (b) Attentive State-space Equition (ASE), and (c) Semantic Guided Neighboring (SGN).

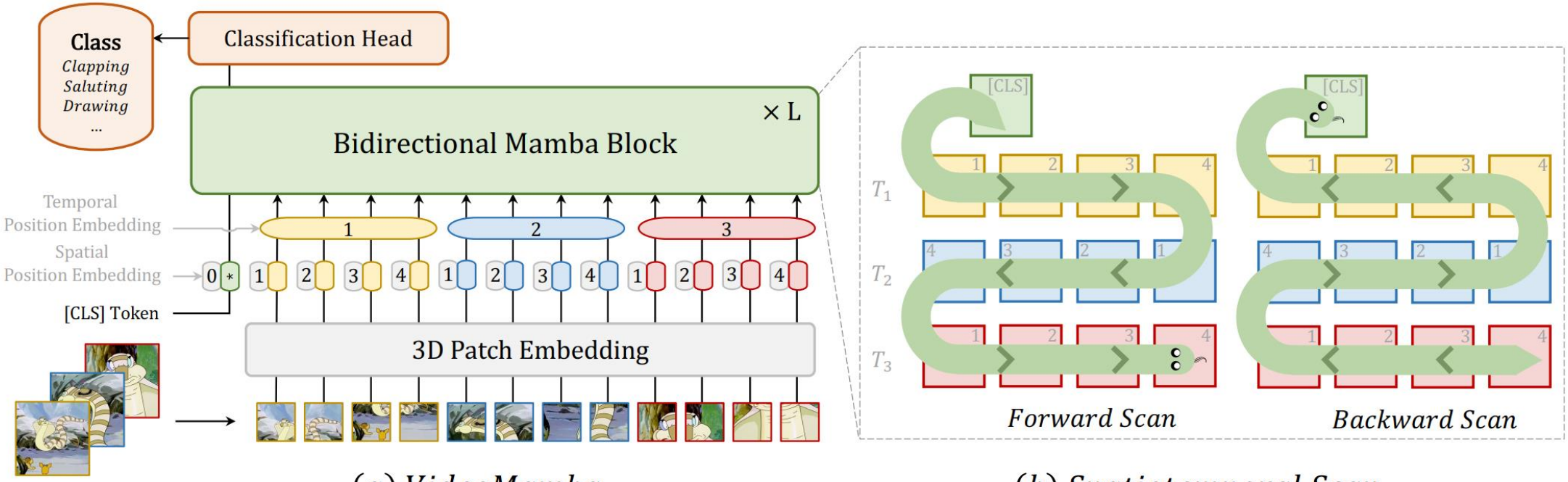
[1] Hang Guo, Yong Guo, Tao Dai, et al. MambaIRv2: Attentive State Space Restoration[C]. 2024 CVPR, **2024/11/22** (cited: 24)

Table 3. Quantitative comparison on ***lightweight image super-resolution*** with state-of-the-art methods. The best and the second best results are in red and blue.

Method	scale	#param	MACs	Set5		Set14		BSDS100		Urban100		Manga109	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
CARN [1]	2×	1,592K	222.8G	37.76	0.9590	33.52	0.9166	32.09	0.8978	31.92	0.9256	38.36	0.9765
LatticeNet [38]	2×	756K	169.5G	38.13	0.9610	33.78	0.9193	32.25	0.9005	32.43	0.9302	-	-
SwinIR-light [33]	2×	910K	244.2G	38.14	0.9611	33.86	0.9206	32.31	0.9012	32.76	0.9340	39.12	0.9783
MambaIR-light [23]	2×	905K	334.2G	38.13	0.9610	33.95	0.9208	32.31	0.9013	32.85	0.9349	39.20	0.9782
ELAN [63]	2×	621K	203.1G	38.17	0.9611	33.94	0.9207	32.30	0.9012	32.76	0.9340	39.11	0.9782
SRFormer-light [70]	2×	853K	236.3G	38.23	0.9613	33.94	0.9209	32.36	0.9019	32.91	0.9353	39.28	0.9785
MambaIRv2-light	2×	774K	286.3G	38.26	0.9615	34.09	0.9221	32.36	0.9019	33.26	0.9378	39.35	0.9785
CARN [1]	3×	1,592K	118.8G	34.29	0.9255	30.29	0.8407	29.06	0.8034	28.06	0.8493	33.50	0.9440
LatticeNet [38]	3×	765K	76.3G	34.53	0.9281	30.39	0.8424	29.15	0.8059	28.33	0.8538	-	-
SwinIR-light [33]	3×	918K	111.2G	34.62	0.9289	30.54	0.8463	29.20	0.8082	28.66	0.8624	33.98	0.9478
MambaIR-light	3×	913K	148.5G	34.63	0.9288	30.54	0.8459	29.23	0.8084	28.70	0.8631	34.12	0.9479
ELAN [63]	3×	629K	90.1G	34.61	0.9288	30.55	0.8463	29.21	0.8081	28.69	0.8624	34.00	0.9478
SRformer-light [70]	3×	861K	105.4G	34.67	0.9296	30.57	0.8469	29.26	0.8099	28.81	0.8655	34.19	0.9489
MambaIRv2-light	3×	781K	126.7G	34.71	0.9298	30.68	0.8483	29.26	0.8098	29.01	0.8689	34.41	0.9497
CARN [1]	4×	1,592K	90.9G	32.13	0.8937	28.60	0.7806	27.58	0.7349	26.07	0.7837	30.47	0.9084
LatticeNet [38]	4×	777K	43.6G	32.30	0.8962	28.68	0.7830	27.62	0.7367	26.25	0.7873	-	-
SwinIR-light [33]	4×	930K	63.6G	32.44	0.8976	28.77	0.7858	27.69	0.7406	26.47	0.7980	30.92	0.9151
MambaIR-light [23]	4×	924K	84.6G	32.42	0.8977	28.74	0.7847	27.68	0.7400	26.52	0.7983	30.94	0.9135
ELAN [63]	4×	640K	54.1G	32.43	0.8975	28.78	0.7858	27.69	0.7406	26.54	0.7982	30.92	0.9150
SRformer-light [70]	4×	873K	62.8G	32.51	0.8988	28.82	0.7872	27.73	0.7422	26.67	0.8032	31.17	0.9165
MambaIRv2-light	4×	790K	75.6G	32.51	0.8992	28.84	0.7878	27.75	0.7426	26.82	0.8079	31.24	0.9182

3. Vision Mamba

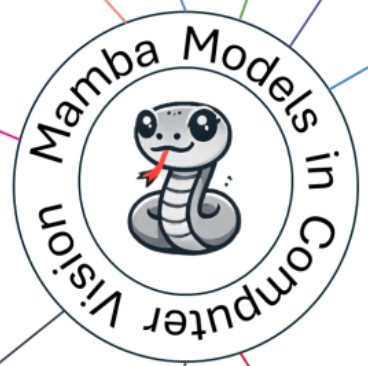
- Video Mamba
  - First to work to adapt state space models for video understanding



(a) *VideoMamba* (b) *Spatiotemporal Scan*

Fig. 3: Framework of VideoMamba. We strictly follow the architecture of vanilla ViT [15], and adapt the bidirectional mamba block [91] for 3D video sequences.

[1] Kunchang Li, Yu Qiao, et al. Videomamba: State space model for efficient video understanding [C]. 2024 ECCV, **2024/03/12** (cited: 274)



Generation and Restoration

DIS | MD-Dose | Gamba | MotionMamba | Zigma | UNITraj | Dim | Dim2 | Dim-3D | Dimba | MambaIR | CU-Mamba | VmambaIR | Serpent | MMA | GMSR

Video Processing

VIS4mer | VideoMamba | Video Mamba Suite | DeMamba | SSM Diffusion | SSSMLV | Matten | RhythmMamba | SSM Event Vision | HARMamba | VideoMambaPro

Image Enhancement

FDVM-Net | UVM-Net | PixMamba | DVMSR | WaterMamba | LFMamba | HDMba | LLEMamba | MambaUIE&SR | FourierMamba | RetinexMamba | BVI-RLV | LLEMamba | MLFSR

Multimodal

SurvMamba | Meteor | TransMA | CMVim | SpikeMba | BoardMamba | Mamba-FETrack | MambaTalk | VL-Mamba | MambaMorph | Sigma | FusionMamba | ReMamber | Cobra | MambaDFuse | TM-Mamba

Classification ,Detection and Segmentation

InsectMamba | Res-VMamba | Fusion-Mamba | MIM-ISTD | SOAR | Mamba-Yolo | HTD-Mamba | RWKV-SAM | Vmamba-CS | Voxel Mamba

Point Cloud Analysis

PointMamba | 3DMambaComplete | 3DMambaIPF | PCM | Mamba3D | Point Mamba | MAMBA4D | OverlapMamba | Mamba24/8D | Mamba24/8D

General Purpose

VMamba | Vision Mamba | Mamba-ND | LocalMamba | EfficientVMamba | PlainMamba | MambaMixer | SiMBA | Heracles | Vim-F | Mamba-R | MSVMamba | FractalVMamba

Medical Image Analysis

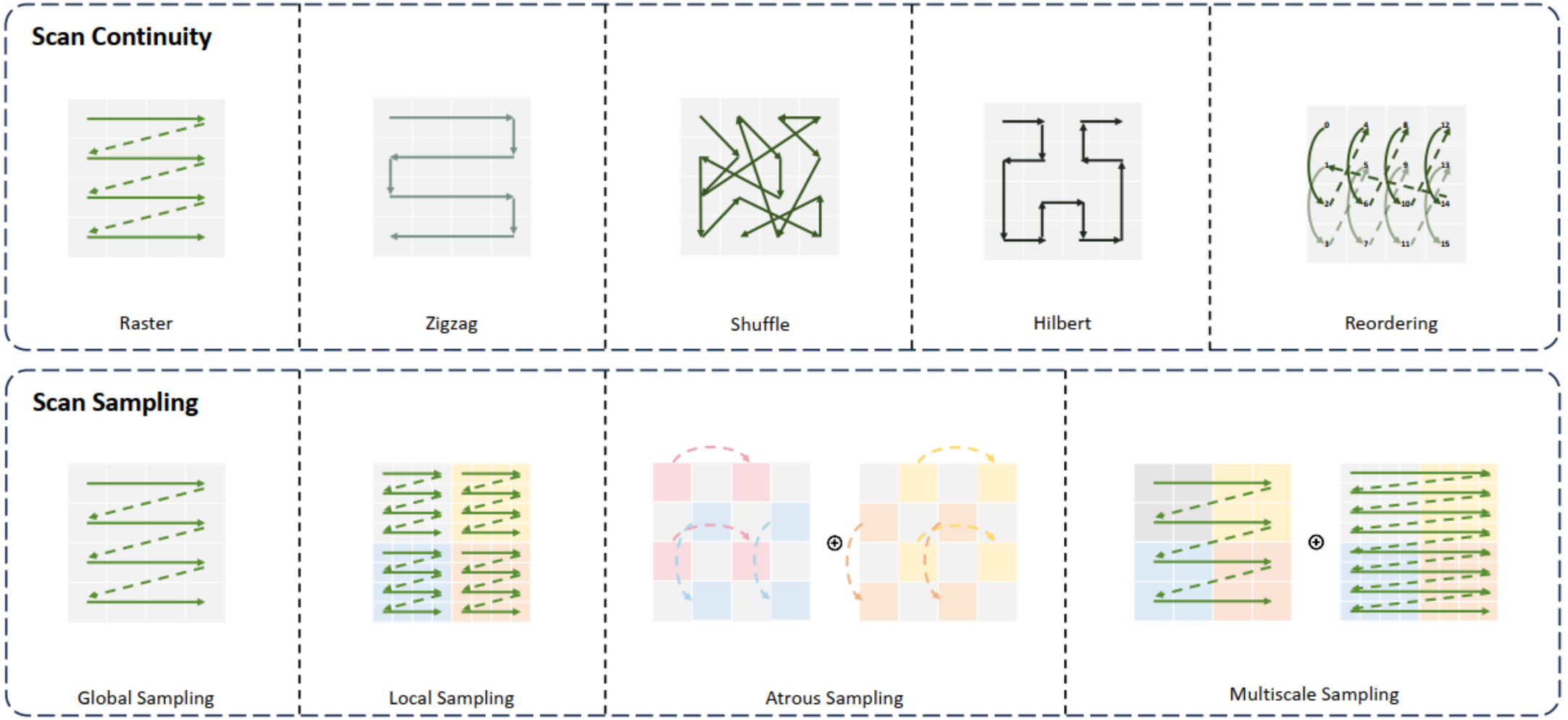
MamMIL | MedMamba | MamabaMIL | Vim4Path | Weak-Mamba-UNet | CMVIM | Vision Mamba | U-Mamba | Mamba-UNet | VM-Unet | BU-Mamba | VM-Unet V2 | H-vmunet | UltraLight VM-UNet | P-Mamba | T-Mamba | Vivim | Semi-Mamba-UNet | ProMamba | MUCM-Net | AC-MambaSeg | VIM-Unet | HC-Mamba | SliceMamba | MHS-VM | XLSTM-UNET | LightM-UNet | LKM-UNet | TM-UNet | Mamba-HUNet | SegMamba | nnMamba | BI-Mamba | I2I-Mamba | TokenUnify | CAF-MambaSegNet | MambaMIR | MambaMIR-GAN | MMR-Mamba | VM-DDPM | Smamba-UNet | Deform-Mamba | SR-Mamba | VMambaMorph | MGDC Tracker | MD-Dose | Swim-UMamba

Remote Sensing

HSIMamba | SpectralMamba | 3DSS-Mamba | RSMamba | SS-Mamba | S2Mamba | HSIDMamba | Pan-Mamba | ChangeMamba | RSCaMa | RS3Mamba | Samba | CM-UNet | LE-Mamba | RSDehamba | FMSR | Seg-LSATM | CDMamba | VMRNN | RS-Mamba

3. Vision Mamba

- Key issue: design a proper scanning strategy for 2D images



目录

4. Mamba2

Insight Model Algorithm

Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality

Theory

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

[1]Department of Computer Science, Princeton University

[2]Machine Learning Department, Carnegie Mellon University
tri@tridao.me, agu@cs.cmu.edu

- 理论层面 (Structured State Space Duality)：统一 Mamba (SSMs) 与 Attention variants (Transformers)
- 算法层面 (Efficient Algorithms)：使用矩阵乘法大幅加速Mamba的训练

[1] **Gu A**, **Dao T**. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality [C]. ICML, 2024/05/31. (cited: 575) (https://tridao.me/blog/2024/mamba2-part1-model/)

4. Mamba2: Problems in Mamba1

- Problem 1 (Understanding)
  - **Mamba范式的理解欠缺**：SSMs 与多种**序列模型范式**有着联系，包括**continuous**, **convolutional**, and **recurrent** sequence models

> *Question 1: What are the conceptual connections between state space models and attention? Can we combine them?*

Yes, SSD Theory

- Problem 2 (Efficiency)
  - **计算速度慢**：Mamba 的计算效率和硬件适配度仍比注意力机制低 (Small FLOPs but **low training speed**)
  - **硬件不适配**：Accelerators such as **GPUs** and **TPUs** are highly specialized for **matrix multiplications**

> *Question 2: Can we speed up the training of Mamba models by recasting them as matrix multiplications?*

Yes, speed up 2-8×

4.1 Mamba2: The Model

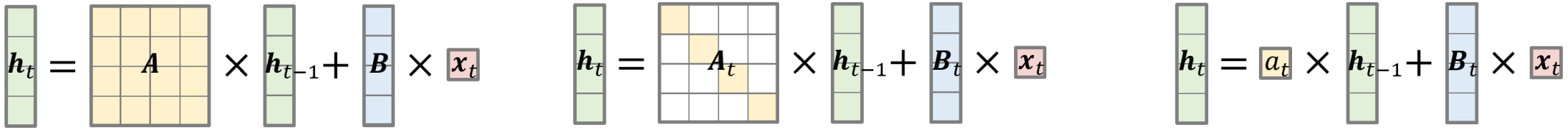
- Mamba2 模型上的改进
  - **Scalar Structured SSMs**：将mamba1中的对角矩阵$\boldsymbol{A}_t$ 进一步的简化为了一个标量 a_t

Mamba1

$$\begin{cases} \boldsymbol{h}_t = \boldsymbol{A}_t \times \boldsymbol{h}_{t-1} + \boldsymbol{B}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

Mamba2

$$\begin{cases} \boldsymbol{h}_t = a_t \times \boldsymbol{h}_{t-1} + \boldsymbol{B}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$



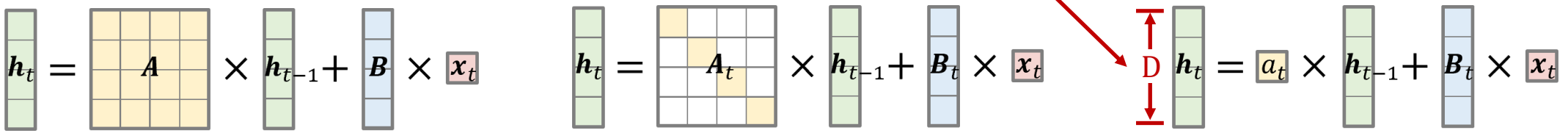
(a) General (Unstructured) SSMs　(b) Mamba1 (Structured SSMs)　(c) Mamba2 (Scalar Structured SSMs)

where $\boldsymbol{x} = [\boldsymbol{x}_0 \quad \boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_{\mathrm{N}-1}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}, \boldsymbol{y} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}, \boldsymbol{h}_t \in \mathbb{R}^{\mathrm{D}\times C}$；　$a_t \in \mathbb{R}$ **(0~1)**, $\boldsymbol{B}_t \in \mathbb{R}^{\mathrm{D}\times 1}, \boldsymbol{C}_t \in \mathbb{R}^{1\times\mathrm{D}}$

  - **Multihead SSMs**：将Mamba1中的单头改为多头，将token $\boldsymbol{x}_t$ 沿着通道维度分为多头，多头之间相互独立（类似于多头自注意力）

4.1 Mamba2: The Model

- Mamba2 模型上的改进
  - **Scalar Structured SSMs**：将mamba1中的对角矩阵$\boldsymbol{A}_t$ 进一步的简化为了一个标量 a_t
  - **Multihead SSMs**：将Mamba1中的单头改为多头，将token $\boldsymbol{x}_t$ 沿着通道维度分为多头，多头之间相互独立（类似于多头自注意力）
  - 做这些改进的**原始动机**：**提高计算效率**（推导出对偶的注意力形式以便使用矩阵乘法）
  - 改进后的**优势**：Mamba-2 allows much larger state dimensions （from D=16 in Mamba-1 to D=64 or 256 or even higher in Mamba-2)



(a) General (Unstructured) SSMs　(b) Mamba1 (Structured SSMs)　(c) Mamba2 (Scalar Structured SSMs)

4.1 Mamba2: The Model

- Mamba2 模型上的改进
  - **Scalar Structured SSMs**：将mamba1中的对角矩阵$\boldsymbol{A}_t$ 进一步的简化为了一个标量 a_t
  - **Multihead SSMs**：将Mamba1中的单头改为多头，将token $\boldsymbol{x}_t$ 沿着通道维度分为多头，多头之间相互独立（类似于多头自注意力）
  - 做这些改进的**原始动机**：**提高计算效率**（推导出对偶的注意力形式以便使用矩阵乘法）
  - 改进后的**优势**：Mamba-2 allows much larger state dimensions （from D=16 in Mamba-1 to D=64 to N=256 or even higher in Mamba-2）
  - **训练速度**: **2-8×** faster during training
  - **模型性能**: Mamba-2 is not strictly better than Mamba-1 (it should generally be on par or better across the board)

4.1 Mamba2: The Model

Table 1: (**Zero-shot Evaluations**.) Best results for each size in bold, second best unlined. We compare against open source LMs with various tokenizers, trained for up to 300B tokens. Pile refers to the validation split, comparing only against models trained on the same dataset and tokenizer (GPT-NeoX-20B). For each model size, Mamba-2 outperforms Mamba, and generally matches Pythia at twice the model size. Full results in Table 10.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	OpenbookQA acc ↑	Average acc ↑
Pythia-1B	NeoX	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	31.4	49.0
Mamba-790M	NeoX	7.33	6.02	**62.7**	**55.1**	**72.1**	**61.2**	**29.5**	56.1	34.2	53.0
Mamba-2-780M	NeoX	**7.26**	**5.86**	61.7	54.9	72.0	61.0	28.5	**60.2**	**36.2**	**53.5**
Hybrid H3-1.3B	GPT2	—	11.25	49.6	52.6	71.3	59.2	28.1	56.9	34.4	50.3
Pythia-1.4B	NeoX	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	30.8	51.7
RWKV4-1.5B	NeoX	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	34.0	51.4
Mamba-1.4B	NeoX	6.80	5.04	65.0	59.1	**74.2**	**65.5**	32.8	**61.5**	36.4	**56.4**
Mamba-2-1.3B	NeoX	**6.66**	**5.02**	**65.7**	**59.9**	73.2	64.3	**33.3**	60.9	**37.8**	**56.4**
Hybrid H3-2.7B	GPT2	—	7.92	55.7	59.7	73.3	65.6	32.3	61.4	33.6	54.5
Pythia-2.8B	NeoX	6.73	5.04	64.7	59.3	74.0	64.1	32.9	59.7	35.2	55.7
RWKV4-3B	NeoX	7.00	5.24	63.9	59.6	73.7	67.8	33.1	59.6	37.0	56.4
Mamba-2.8B	NeoX	6.22	4.23	69.2	66.1	75.2	**69.7**	36.3	63.5	**39.6**	59.9
Mamba-2-2.7B	NeoX	**6.09**	**4.10**	**69.7**	**66.6**	**76.4**	69.6	**36.4**	**64.0**	38.8	**60.2**

4.2 Mamba2: The Theory

- Mamba2 状态空间方程的两种**对偶**形式
  - **循环递推**形式（Recurrent Form: 串行计算，速度慢）

$$\begin{cases} \boldsymbol{h}_t = a_t \times \boldsymbol{h}_{t-1} + \boldsymbol{B}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

  - **注意力**形式（Attention Form: 矩阵乘法，支持并行计算，速度快）

$$\boldsymbol{y}_0 = \boldsymbol{C}_0\, \boldsymbol{B}_0 \boldsymbol{x}_0$$

$$\boldsymbol{y}_1 = \boldsymbol{C}_1\, a_1 \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_1 \boldsymbol{B}_1 \boldsymbol{x}_1$$

$$\boldsymbol{y}_2 = \boldsymbol{C}_2 a_2 a_1 \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_2 a_2 \boldsymbol{B}_1 \boldsymbol{x}_1 + \boldsymbol{C}_2 \boldsymbol{B}_2 \boldsymbol{x}_2$$

$$\dots$$

$$\boldsymbol{y}_t = \boldsymbol{C}_t (a_t \cdots a_1) \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_t (a_t \cdots a_2) \boldsymbol{B}_1 \boldsymbol{x}_1 + \cdots + \boldsymbol{C}_t a_t\, \boldsymbol{B}_{t-1} \boldsymbol{x}_{t-1} + \boldsymbol{C}_t \boldsymbol{B}_t \boldsymbol{x}_t$$

- **循环递推**形式（Recurrent Form: 串行计算，速度慢）

$$\begin{cases} \boldsymbol{h}_t = a_t \times \boldsymbol{h}_{t-1} + \boldsymbol{B}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

- **注意力**形式（Attention Form: 矩阵乘法，支持并行计算，速度快）

$$\boldsymbol{y}_0 = \boldsymbol{C}_0\, \boldsymbol{B}_0 \boldsymbol{x}_0$$

$$\boldsymbol{y}_1 = \boldsymbol{C}_1\, a_1 \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_1 \boldsymbol{B}_1 \boldsymbol{x}_1$$

$$\boldsymbol{y}_2 = \boldsymbol{C}_2 a_2 a_1 \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_2 a_2 \boldsymbol{B}_1 \boldsymbol{x}_1 + \boldsymbol{C}_2 \boldsymbol{B}_2 \boldsymbol{x}_2$$

$$\cdots$$

$$\boldsymbol{y}_t = \boldsymbol{C}_t (a_t \cdots a_1) \boldsymbol{B}_0 \boldsymbol{x}_0 + \boldsymbol{C}_t (a_t \cdots a_2) \boldsymbol{B}_1 \boldsymbol{x}_1 + \cdots + \boldsymbol{C}_t a_t\, \boldsymbol{B}_{t-1} \boldsymbol{x}_{t-1} + \boldsymbol{C}_t \boldsymbol{B}_t \boldsymbol{x}_t$$

$$\Rightarrow \underbrace{\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{C}}} = \underbrace{\begin{bmatrix} \boldsymbol{C}_0 \boldsymbol{B}_0^\top & 0 & \cdots & 0 & 0 \\ \boldsymbol{C}_1 a_1 \boldsymbol{B}_0^\top & \boldsymbol{C}_1 A_{1:1} \boldsymbol{B}_1^\top & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}(a_{N-2} \cdots a_1) \boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}(a_{N-2} \cdots a_2) \boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-2} \boldsymbol{B}_{N-2}^\top & 0 \\ \boldsymbol{C}_{N-1}(a_{N-1} \cdots a_1) \boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}(a_{N-1} \cdots a_2) \boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-1} a_{N-1} \boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1} \boldsymbol{B}_{N-1}^\top \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{N}}} \underbrace{\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{C}}}$$

- **注意力**形式（Attention Form: 矩阵乘法，支持并行计算，速度快）

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & 0 & \dots & 0 & 0 \\ \boldsymbol{C}_1 a_1 \boldsymbol{B}_0^\top & \boldsymbol{C}_1 A_{1:1} \boldsymbol{B}_1^\top & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}(a_{N-2}\cdots a_1)\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}(a_{N-2}\cdots a_2)\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & 0 \\ \boldsymbol{C}_{N-1}(a_{N-1}\cdots a_1)\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}(a_{N-1}\cdots a_2)\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-1}a_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$

- 令 $A_{i:j} = \begin{cases} a_{i+1} \times \cdots \times a_j & if\ i < j \\ 1 & if\ i = j \\ a_{j+1} \times \cdots \times a_i & if\ i > j \end{cases}$

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_0 A_{0:0}\, \boldsymbol{B}_0^\top & 0 & \dots & 0 & 0 \\ \boldsymbol{C}_1 A_{1:0} \boldsymbol{B}_0^\top & \boldsymbol{C}_1 A_{1:1} \boldsymbol{B}_1^\top & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2} A_{N-2:0} \boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2} A_{N-2:1} \boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-2} A_{N-2:N-2} \boldsymbol{B}_{N-2}^\top & 0 \\ \boldsymbol{C}_{N-1} A_{N-1:0} \boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1} A_{N-1:1} \boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-1} A_{N-1:N-2} \boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1} A_{N-1:N-1} \boldsymbol{B}_{N-1}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & \boldsymbol{C}_0\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_1\boldsymbol{B}_0^\top & \boldsymbol{C}_1\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^\top & \dots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} A_{0:0} & 0 & \dots & 0 & 0 \\ A_{1:0} & A_{1:1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \dots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \dots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$

- **The Paradigm of Self-attention**
  - For input tokens $\boldsymbol{x} = [\boldsymbol{x}_0 \quad \boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_{\mathrm{N}-1}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$, output tokens $\boldsymbol{y} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$ is formulated as:

$$\boldsymbol{Q} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_Q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{K} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_K \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{V} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_V \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}},$$

$$\boldsymbol{Y} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \times \boldsymbol{V} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$$

Complexity: $\mathcal{O}(N^2)$

- **The Paradigm of Vanilla Linear Attention**

$$\boldsymbol{Y} = (\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \times \boldsymbol{V} = \boldsymbol{Q} \times (\boldsymbol{K}^{\mathsf{T}}\boldsymbol{V})$$

Complexity: $\mathcal{O}(N)$

- **The Paradigm of Causal Linear Attention**[1]

$$\boldsymbol{Y} = \left((\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \odot \boldsymbol{L}\right) \times \boldsymbol{V} = \boldsymbol{Q} \star \mathrm{cumsum}(\boldsymbol{K} \star \boldsymbol{V})$$

Complexity: $\mathcal{O}(N)$

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{K} \star \boldsymbol{V} = \mathrm{einsum}[\mathrm{ND}, \mathrm{NC} \rightarrow \mathrm{NDC}](\boldsymbol{K}, \boldsymbol{V})$$
$$\boldsymbol{\mathcal{H}} = \mathrm{cumsum}(\boldsymbol{K} \star \boldsymbol{V}) = \mathrm{cumsum}(\boldsymbol{\mathcal{Z}}) = [\boldsymbol{\mathcal{Z}}_0 \quad \boldsymbol{\mathcal{Z}}_0 + \boldsymbol{\mathcal{Z}}_1 \quad \cdots \quad \textstyle\sum_{i=0}^{N-1} \boldsymbol{\mathcal{Z}}_i]$$
$$\boldsymbol{Y} = \boldsymbol{Q} \star \boldsymbol{\mathcal{H}} = \mathrm{einsum}[\mathrm{ND}, \mathrm{NDC} \rightarrow \mathrm{NC}](\boldsymbol{Q}, \boldsymbol{\mathcal{H}})$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 & 1 \end{bmatrix}$$

[1] Katharopoulos A, Vyas A, Pappas N, et al. Transformers are rnns: Fast autoregressive transformers with linear attention[C]. 2020 ICML (cited: 2181)

- **The Paradigm of Self-attention**
  - For input tokens $\boldsymbol{x} = [\boldsymbol{x}_0 \quad \boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_{\mathrm{N}-1}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$, output tokens $\boldsymbol{y} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}}$ is formulated as:

$$\begin{aligned}\boldsymbol{Q} &= \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_Q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},\\ \boldsymbol{K} &= \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_K \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},\\ \boldsymbol{V} &= \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_V \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}},\end{aligned} \qquad \boldsymbol{Y} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \times \boldsymbol{V} \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}} \qquad \textit{Complexity: } \mathcal{O}(N^2)$$

- **The Paradigm of Vanilla Linear Attention**

$$\boldsymbol{Y} = (\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \times \boldsymbol{V} = \boldsymbol{Q} \times (\boldsymbol{K}^{\mathsf{T}}\boldsymbol{V}) \qquad \textit{Complexity: } \mathcal{O}(N)$$

- **The Paradigm of Causal Linear Attention**[1]

$$\boldsymbol{Y} = \left((\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}) \odot \boldsymbol{L}\right) \times \boldsymbol{V} = \boldsymbol{Q} \star \mathrm{cumsum}(\boldsymbol{K} \star \boldsymbol{V}) \qquad \textit{Complexity: } \mathcal{O}(N)$$

$$\begin{bmatrix}\boldsymbol{y}_0\\ \boldsymbol{y}_1\\ \vdots\\ \boldsymbol{y}_{N-2}\\ \boldsymbol{y}_{N-1}\end{bmatrix} = \left(\begin{bmatrix}\boldsymbol{Q}_0\boldsymbol{K}_0^{\mathsf{T}} & \boldsymbol{Q}_0\boldsymbol{K}_1^{\mathsf{T}} & \cdots & \boldsymbol{Q}_0\boldsymbol{K}_{N-2}^{\mathsf{T}} & \boldsymbol{Q}_0\boldsymbol{K}_{N-1}^{\mathsf{T}}\\ \boldsymbol{Q}_1\boldsymbol{K}_0^{\mathsf{T}} & \boldsymbol{Q}_1\boldsymbol{K}_1^{\mathsf{T}} & \cdots & \boldsymbol{Q}_1\boldsymbol{K}_{N-2}^{\mathsf{T}} & \boldsymbol{Q}_1\boldsymbol{K}_{N-1}^{\mathsf{T}}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \boldsymbol{Q}_{N-2}\boldsymbol{K}_0^{\mathsf{T}} & \boldsymbol{Q}_{N-2}\boldsymbol{K}_1^{\mathsf{T}} & \cdots & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-2}^{\mathsf{T}} & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-1}^{\mathsf{T}}\\ \boldsymbol{Q}_{N-1}\boldsymbol{K}_0^{\mathsf{T}} & \boldsymbol{Q}_{N-1}\boldsymbol{K}_1^{\mathsf{T}} & \cdots & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-2}^{\mathsf{T}} & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-1}^{\mathsf{T}}\end{bmatrix} \odot \begin{bmatrix}1 & 0 & \cdots & 0 & 0\\ 1 & 1 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 1 & 1 & \cdots & 1 & 0\\ 1 & 1 & \cdots & 1 & 1\end{bmatrix}\right)\begin{bmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \vdots\\ \boldsymbol{V}_{N-2}\\ \boldsymbol{V}_{N-1}\end{bmatrix}$$

- Causal Linear Attention

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{Q}_0\boldsymbol{K}_0^\top & \boldsymbol{Q}_0\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_0\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_0\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_1\boldsymbol{K}_0^\top & \boldsymbol{Q}_1\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_1\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_1\boldsymbol{K}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{Q}_{N-2}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_{N-1}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{V}_{N-2} \\ \boldsymbol{V}_{N-1} \end{bmatrix}$$

$$\boldsymbol{Q} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_Q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{K} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_K \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{V} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_V \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}},$$

- Mamba2 的注意力形式

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & \boldsymbol{C}_0\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_1\boldsymbol{B}_0^\top & \boldsymbol{C}_1\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$

$$\boldsymbol{B}_t = \Delta_t \times \mathrm{nn.Linear}(\boldsymbol{x}_t) = \Delta_t \times \boldsymbol{x}_t\boldsymbol{W}_B \in \mathbb{R}^{1\times\mathrm{D}}$$
$$\boldsymbol{C}_t = \mathrm{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t\boldsymbol{W}_C \in \mathbb{R}^{1\times\mathrm{D}}$$

$$a = -\exp\left(\mathrm{nn.Paramter}(torch.diag(1))\right) \in \mathbb{R}^1 \ (< 0)$$
$$\Delta_t = \mathrm{Softplus}(\mathrm{MLP}_\Delta(\boldsymbol{x}_t)) \in \mathbb{R}^1 \ (> 0)$$
$$a_t = e^{\Delta_t a} = \exp(\Delta_t a) \in \mathbb{R}^1 \ \mathbf{(0\sim1)}$$

- Causal Linear Attention

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{Q}_0\boldsymbol{K}_0^\top & \boldsymbol{Q}_0\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_0\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_0\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_1\boldsymbol{K}_0^\top & \boldsymbol{Q}_1\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_1\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_1\boldsymbol{K}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{Q}_{N-2}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_{N-1}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{V}_{N-2} \\ \boldsymbol{V}_{N-1} \end{bmatrix}$$

$$\boldsymbol{Q} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_Q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{K} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_K \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}},$$
$$\boldsymbol{V} = \mathrm{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_V \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}},$$

- Mamba2 的注意力形式

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & \boldsymbol{C}_0\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_1\boldsymbol{B}_0^\top & \boldsymbol{C}_1\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \right) \begin{bmatrix} \Delta_0\boldsymbol{x}_0 \\ \Delta_1\boldsymbol{x}_1 \\ \vdots \\ \Delta_{N-2}\boldsymbol{x}_{N-2} \\ \Delta_{N-1}\boldsymbol{x}_{N-1} \end{bmatrix}$$

$$\boldsymbol{B}_t = \mathrm{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t\boldsymbol{W}_B \in \mathbb{R}^{1\times\mathrm{D}}$$
$$\boldsymbol{C}_t = \mathrm{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t\boldsymbol{W}_C \in \mathbb{R}^{1\times\mathrm{D}}$$

$$a = -\exp\left(\mathrm{nn.Paramter}(torch.diag(1))\right) \in \mathbb{R}^1 \ (< 0)$$
$$\Delta_t = \mathrm{Softplus}(\mathrm{MLP}_\Delta(\boldsymbol{x}_t)) \in \mathbb{R}^1 \ (> 0)$$
$$a_t = e^{\Delta_t a} = \exp(\Delta_t a) \in \mathbb{R}^1 \ (\mathbf{0\sim1})$$

- Causal Linear Attention

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{Q}_0\boldsymbol{K}_0^\top & \boldsymbol{Q}_0\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_0\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_0\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_1\boldsymbol{K}_0^\top & \boldsymbol{Q}_1\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_1\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_1\boldsymbol{K}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{Q}_{N-2}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-2}\boldsymbol{K}_{N-1}^\top \\ \boldsymbol{Q}_{N-1}\boldsymbol{K}_0^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_1^\top & \cdots & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-2}^\top & \boldsymbol{Q}_{N-1}\boldsymbol{K}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{V}_{N-2} \\ \boldsymbol{V}_{N-1} \end{bmatrix}$$

$$\boldsymbol{Q} = \text{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_Q \in \mathbb{R}^{\text{N}\times\text{D}},$$

$$\boldsymbol{K} = \text{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_K \in \mathbb{R}^{\text{N}\times\text{D}},$$

$$\boldsymbol{V} = \text{nn.Linear}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}_V \in \mathbb{R}^{\text{N}\times\text{C}},$$

- Mamba2 的注意力形式

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & \boldsymbol{C}_0\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_1\boldsymbol{B}_0^\top & \boldsymbol{C}_1\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$

$$\boldsymbol{B}_t = \text{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t\boldsymbol{W}_B \in \mathbb{R}^{1\times\text{D}}$$

$$\boldsymbol{C}_t = \text{nn.Linear}(\boldsymbol{x}_t) = \boldsymbol{x}_t\boldsymbol{W}_C \in \mathbb{R}^{1\times\text{D}}$$

$$a = -\exp\left(\text{nn.Paramter}(torch.diag(1))\right) \in \mathbb{R}^1 \ (< 0)$$

$$\Delta_t = \text{Softplus}(\text{MLP}_\Delta(\boldsymbol{x}_t)) \in \mathbb{R}^1 \ (> 0)$$

$$a_t = e^{\Delta_t a} = \exp(\Delta_t a) \in \mathbb{R}^1 \ \mathbf{(0\sim1)}$$

4.2 Mamba2: The Theory

- Causal Linear Attention

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \left(\begin{bmatrix} Q_0K_0^\top & Q_0K_1^\top & \cdots & Q_0K_{N-2}^\top & Q_0K_{N-1}^\top \\ Q_1K_0^\top & Q_1K_1^\top & \cdots & Q_1K_{N-2}^\top & Q_1K_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Q_{N-2}K_0^\top & Q_{N-2}K_1^\top & \cdots & Q_{N-2}K_{N-2}^\top & Q_{N-2}K_{N-1}^\top \\ Q_{N-1}K_0^\top & Q_{N-1}K_1^\top & \cdots & Q_{N-1}K_{N-2}^\top & Q_{N-1}K_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{N-2} \\ V_{N-1} \end{bmatrix}$$

- Mamba2 的注意力形式

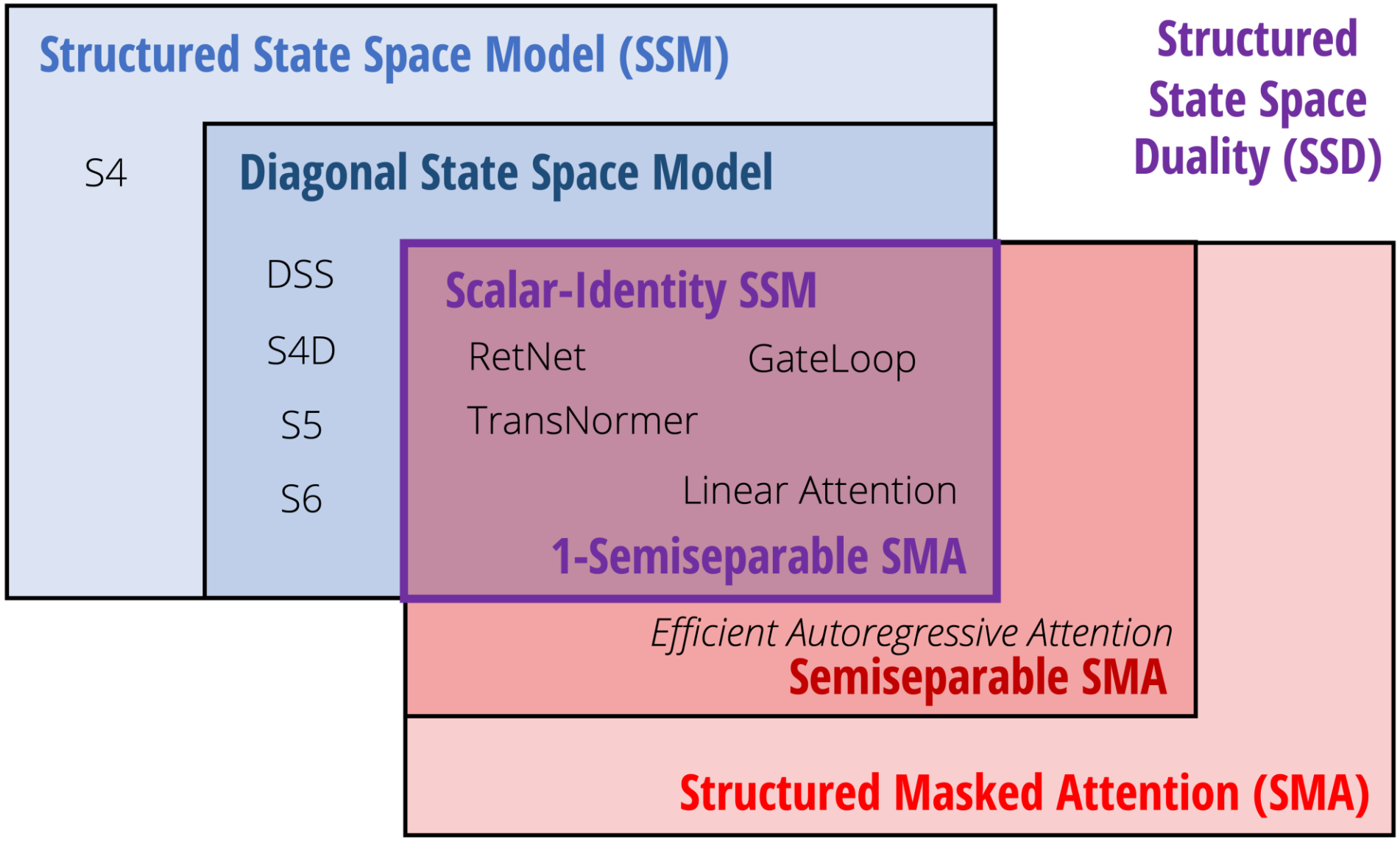
$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \left(\begin{bmatrix} C_0B_0^\top & C_0B_1^\top & \cdots & C_0B_{N-2}^\top & C_0B_{N-1}^\top \\ C_1B_0^\top & C_1B_1^\top & \cdots & C_1B_{N-2}^\top & C_1B_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{N-2}B_0^\top & C_{N-2}B_1^\top & \cdots & C_{N-2}B_{N-2}^\top & C_{N-2}B_{N-1}^\top \\ C_{N-1}B_0^\top & C_{N-1}B_1^\top & \cdots & C_{N-1}B_{N-2}^\top & C_{N-1}B_{N-1}^\top \end{bmatrix} \odot \begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \right) \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}$$

结论：Mamba和Linear Attention本质上是统一的

⬇

Theory: Structured State Space Duality (SSD)

Theory: Structured State Space Duality (SSD)



Theory: Structured State Space Duality (SSD)

Structured State Space Model		Structured Masked Attention	
C	(contraction matrix)	Q	(queries)
B	(expansion matrix)	K	(keys)
X	(input sequence)	V	(values)
$A_{j:i}$	(state matrix)	L_{ji}	(mask)
N	(state expansion dim.)	N	(kernel feature dim.)
H	(hidden states (8b))	SMA linear dual (15)	
$= L \cdot XB$	(linear mode)		
SSM quadratic dual (16)		G	(Gram matrix (13a))
		$= Q \cdot K^\top$	(quadratic mode)

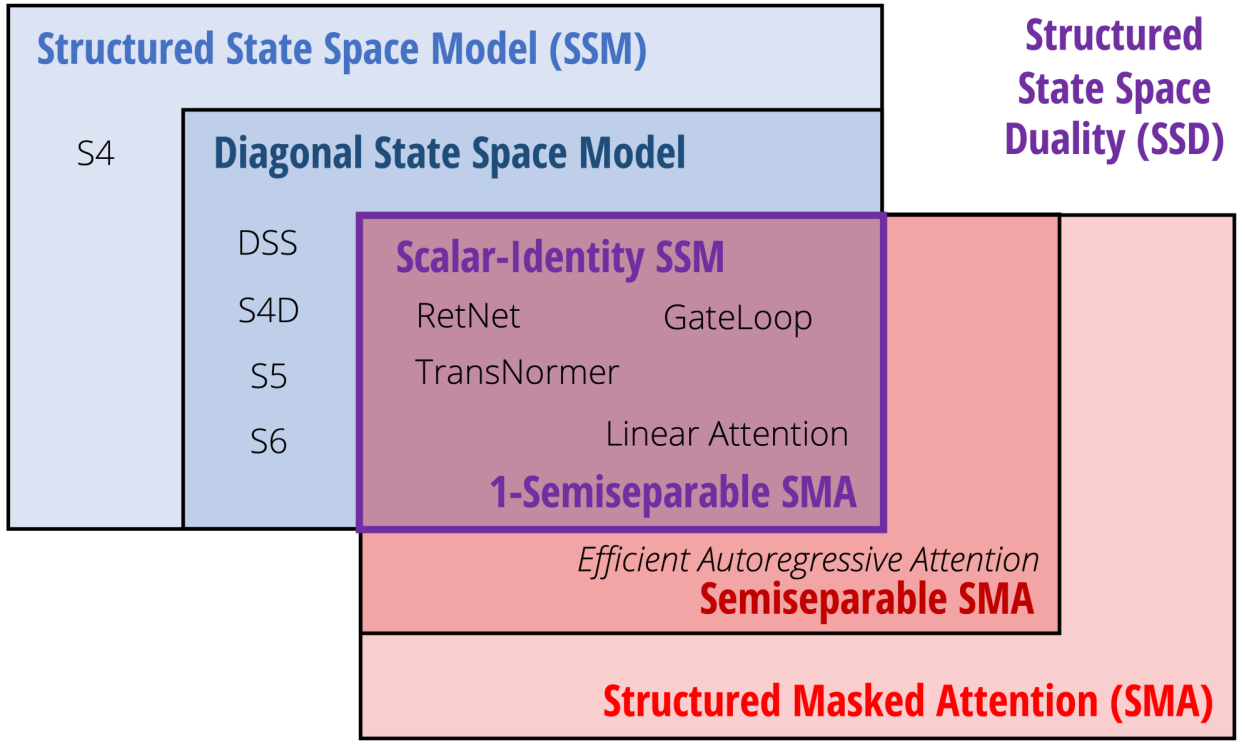
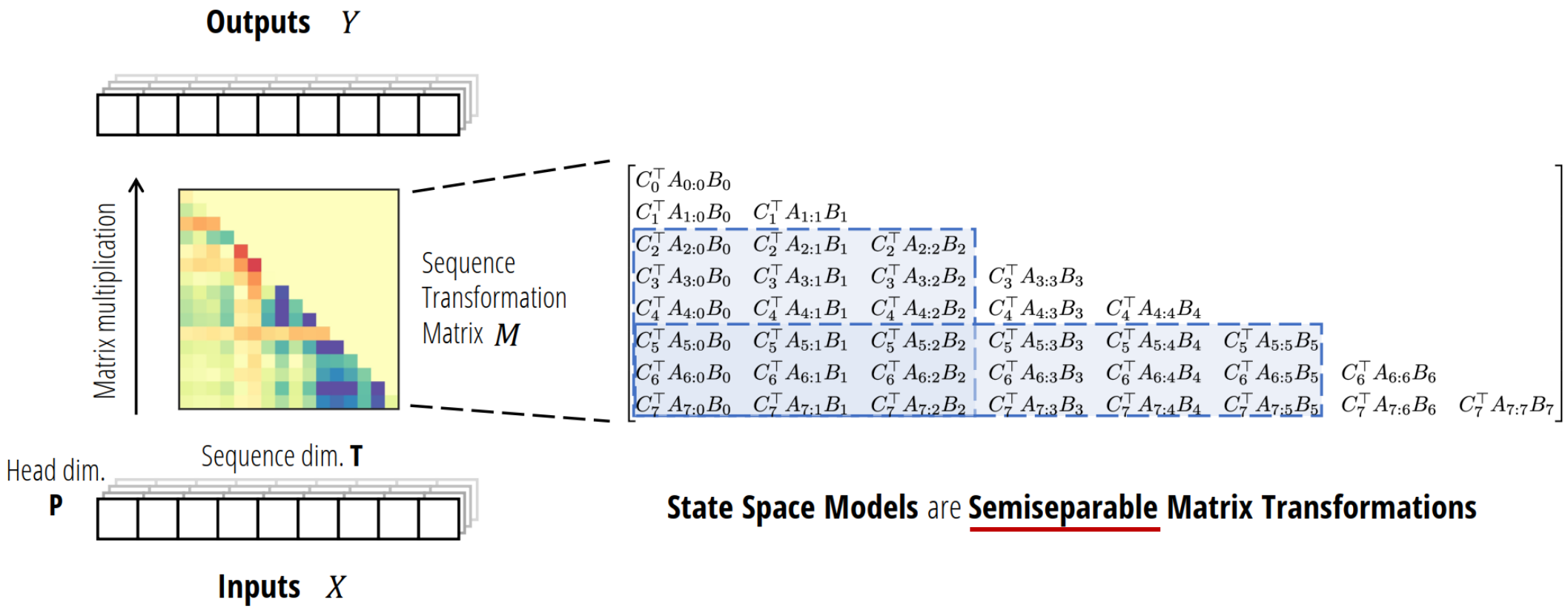


Figure 4: (**Structured State Space Duality**.) State space duality describes the close relationship between state space models and masked attention. (*Left*) General SSMs and SMA both possess linear and quadratic forms, with direct analogs in notation. (*Right*) SSMs and SMA intersect at a large class of *state space dual models* (SSD) which capture many sequence models as special cases.

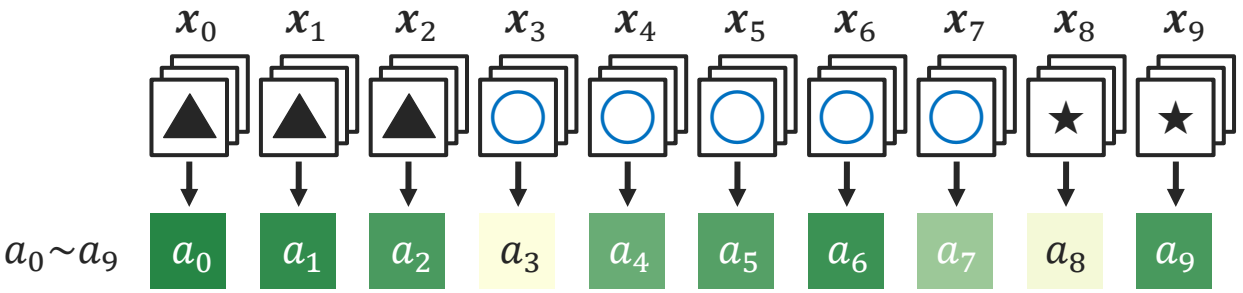
- Mamba2: Structured Masked Attention

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \Bigg(\underbrace{\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^\top & \boldsymbol{C}_0\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_1\boldsymbol{B}_0^\top & \boldsymbol{C}_1\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^\top \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^\top & \cdots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^\top & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^\top \end{bmatrix}}_{\text{Linear Attention}} \odot \underbrace{\begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix}}_{\text{Structured Mask } \boldsymbol{L}} \Bigg) \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}$$



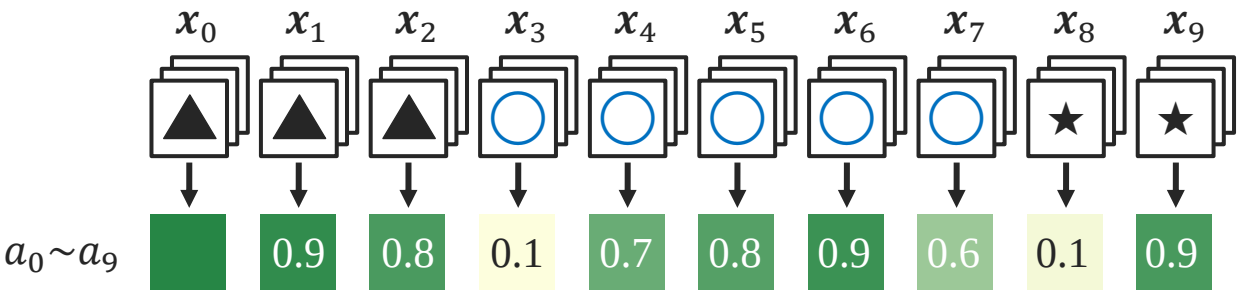
State Space Models are **Semiseparable** Matrix Transformations

- Structured Mask $\boldsymbol{L}$
  - 1-semiseparable structured matrix
  - 衰减因子 a_t 介于 **(0~1)**
  - $L_{i,j} = a_{i+1} \times \cdots \times a_j$ (即$\boldsymbol{x}_i$到$\boldsymbol{y}_j$**途径**所有衰减因子的连乘)
  - 可以理解为一种 input-dependent **relative positional encodings**



$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ a_1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-2} \cdots a_1 & a_{N-2} \cdots a_2 & \dots & 1 & 0 \\ a_{N-1} \cdots a_1 & a_{N-1} \cdots a_2 & \dots & a_{N-1} & 1 \end{bmatrix}$$

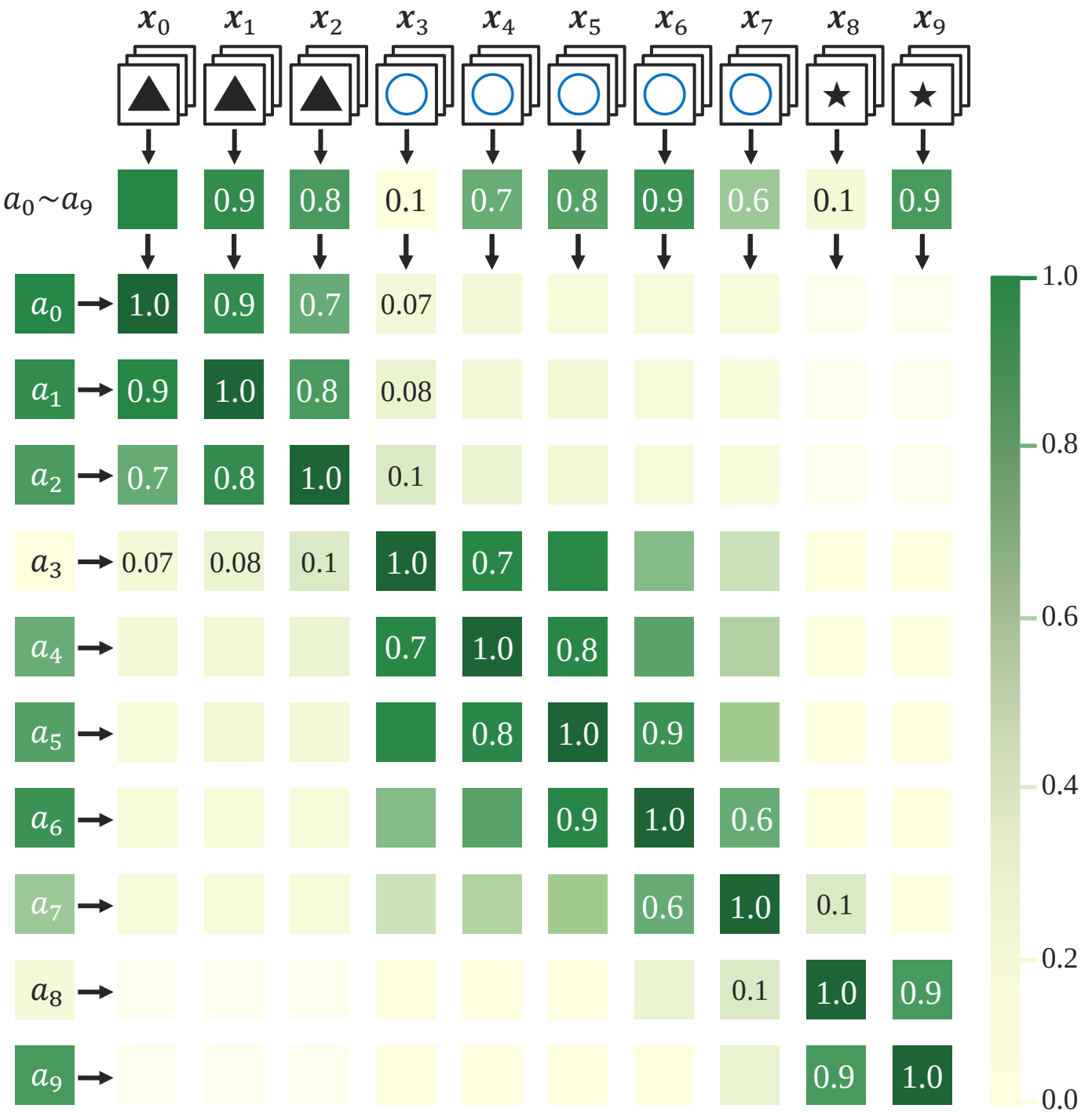
- Structured Mask $\boldsymbol{L}$
  - 1-semiseparable structured matrix
  - 衰减因子 a_t 介于 **(0~1)**
  - $L_{i,j} = a_{i+1} \times \cdots \times a_j$ (即$\boldsymbol{x}_i$到$\boldsymbol{y}_j$**途径**所有衰减因子的连乘)
  - 可以理解为一种 input-dependent **relative positional encodings**



$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ a_1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-2} \cdots a_1 & a_{N-2} \cdots a_2 & \dots & 1 & 0 \\ a_{N-1} \cdots a_1 & a_{N-1} \cdots a_2 & \dots & a_{N-1} & 1 \end{bmatrix}$$

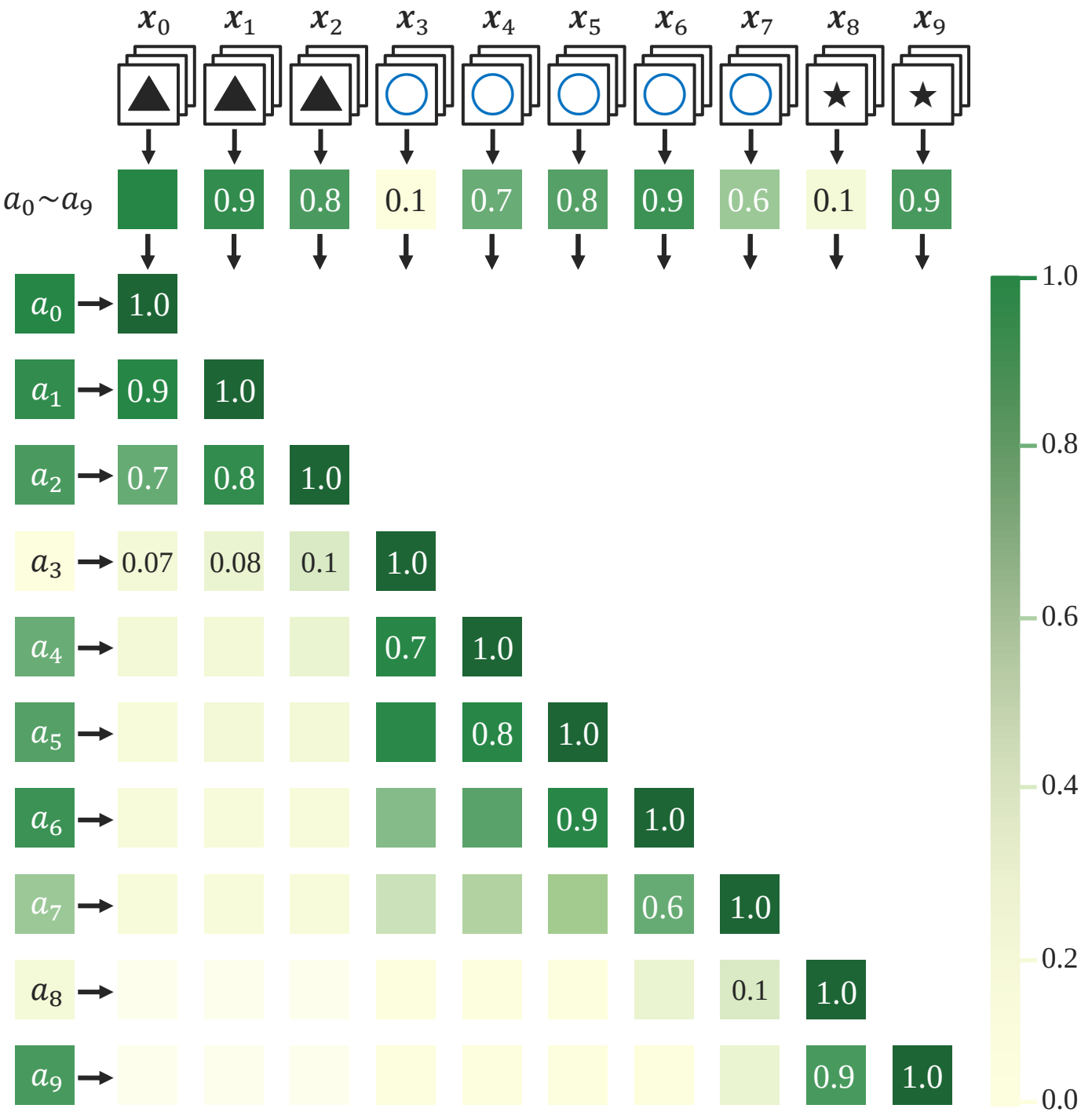
- Structured Mask $\boldsymbol{L}$
  - 1-semiseparable structured matrix
  - 衰减因子 a_t 介于 **(0~1)**
  - $L_{i,j} = a_{i+1} \times \cdots \times a_j$ (即$\boldsymbol{x}_i$到$\boldsymbol{y}_j$**途径**所有衰减因子的连乘)
  - 可以理解为一种 input-dependent **relative positional encodings**

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ a_1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-2} \cdots a_1 & a_{N-2} \cdots a_2 & \dots & 1 & 0 \\ a_{N-1} \cdots a_1 & a_{N-1} \cdots a_2 & \dots & a_{N-1} & 1 \end{bmatrix}$$



- Structured Mask $\boldsymbol{L}$
  - 1-semiseparable structured matrix
  - 衰减因子 a_t 介于 **(0~1)**
  - $L_{i,j} = a_{i+1} \times \cdots \times a_j$ (即$\boldsymbol{x}_i$到$\boldsymbol{y}_j$**途径**所有衰减因子的连乘)
  - 可以理解为一种 input-dependent **relative positional encodings**

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ a_1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-2} \cdots a_1 & a_{N-2} \cdots a_2 & \dots & 1 & 0 \\ a_{N-1} \cdots a_1 & a_{N-1} \cdots a_2 & \dots & a_{N-1} & 1 \end{bmatrix}$$



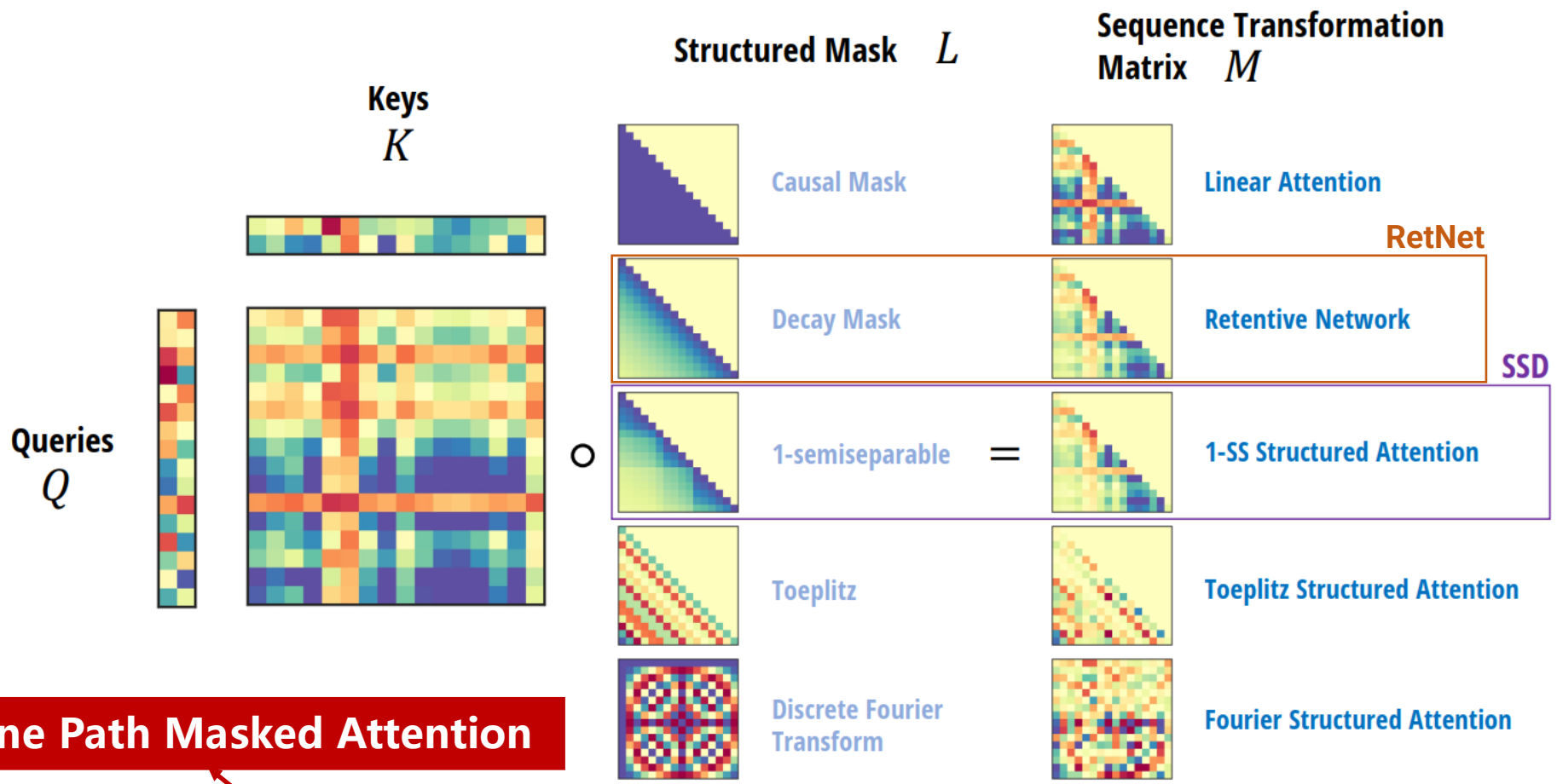


Figure 3: (**Structured Masked Attention**.) SMA constructs a masked attention matrix $M = QK^\top \circ L$ for any structured matrix L, which defines a matrix sequence transformation $Y = MV$. All instances of SMA have a dual subquadratic form induced by a different contraction ordering, combined with the efficient structured matrix multiplication by L. Previous examples include Linear Attention (Katharopoulos et al. 2020) and RetNet (Y. Sun et al. 2023). Beyond SSD (1-semiseparable SMA), the focus of this paper, many other potential instantiations of structured attention are possible.

- The decay mask could be generalized to a Toeplitz matrix $L_{ij} = \alpha_{i-j}$ for some learnable (or input-dependent) set of parameters $\alpha \in \mathbb{R}^\top$. This can be interpreted as a form of relative positional encoding, reminiscent of other methods such as AliBi (Press, N. Smith, and Lewis 2022) but multiplicative instead of additive.

4.3 Mamba2: The Algorithm

- Mamba2的两种对偶形式
  - **循环递推**形式（**串行计算，速度慢**） *Complexity:* $\mathcal{O}(N)$

$$\begin{cases} \boldsymbol{h}_t = a_t \times \boldsymbol{h}_{t-1} + \boldsymbol{B}_t \times \boldsymbol{x}_t \\ \boldsymbol{y}_t = \boldsymbol{C}_t \times \boldsymbol{h}_t \end{cases}$$

  - Structured Masked Attention形式（**并行计算，矩阵乘法，速度快**） *Complexity:* $\mathcal{O}(N^2)$ ✕

$$\boldsymbol{Y} = \boldsymbol{M}\boldsymbol{X} = (\boldsymbol{C}\boldsymbol{B}^{\top} \odot \boldsymbol{L})\boldsymbol{X}$$

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-2} \\ \boldsymbol{y}_{N-1} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{C}}} = \left(\underbrace{\begin{bmatrix} \boldsymbol{C}_0\boldsymbol{B}_0^{\top} & \boldsymbol{C}_0\boldsymbol{B}_1^{\top} & \cdots & \boldsymbol{C}_0\boldsymbol{B}_{N-2}^{\top} & \boldsymbol{C}_0\boldsymbol{B}_{N-1}^{\top} \\ \boldsymbol{C}_1\boldsymbol{B}_0^{\top} & \boldsymbol{C}_1\boldsymbol{B}_1^{\top} & \cdots & \boldsymbol{C}_1\boldsymbol{B}_{N-2}^{\top} & \boldsymbol{C}_1\boldsymbol{B}_{N-1}^{\top} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{N-2}\boldsymbol{B}_0^{\top} & \boldsymbol{C}_{N-2}\boldsymbol{B}_1^{\top} & \cdots & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-2}^{\top} & \boldsymbol{C}_{N-2}\boldsymbol{B}_{N-1}^{\top} \\ \boldsymbol{C}_{N-1}\boldsymbol{B}_0^{\top} & \boldsymbol{C}_{N-1}\boldsymbol{B}_1^{\top} & \cdots & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-2}^{\top} & \boldsymbol{C}_{N-1}\boldsymbol{B}_{N-1}^{\top} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{N}}} \odot \underbrace{\begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{N}}} \right) \underbrace{\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N-2} \\ \boldsymbol{x}_{N-1} \end{bmatrix}}_{\mathbb{R}^{\mathbf{N}\times\mathbf{C}}}$$

$\mathcal{O}(N^2)$ $\mathcal{O}(N^2)$

Theorem 1. *For any matrices* $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{N\times D}, \boldsymbol{L} \in \mathbb{R}^{N\times N}, \boldsymbol{V} \in \mathbb{R}^{N\times C}$, *the following equation holds:*

$$\boldsymbol{Y} = (\boldsymbol{Q}\boldsymbol{K}^{\top} \odot \boldsymbol{L})\boldsymbol{V} \quad \Leftrightarrow$$

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{K} \star \boldsymbol{V} = \text{einsum}[\text{ND}, \text{NC} \rightarrow \text{NDC}](\boldsymbol{K}, \boldsymbol{V}) \qquad \mathcal{O}(NDC)$$

$$\boldsymbol{\mathcal{H}} = \text{einsum}[\text{MN}, \text{NDC} \rightarrow \text{MDC}](\boldsymbol{L}, \boldsymbol{\mathcal{Z}}), \quad (\text{M} = \text{N}) \qquad \mathcal{O}(N^2DC)$$

$$\boldsymbol{Y} = \boldsymbol{Q} \star \boldsymbol{\mathcal{H}} = \text{einsum}[\text{MD}, \text{MDC} \rightarrow \text{MC}](\boldsymbol{Q}, \boldsymbol{\mathcal{H}}) \qquad \mathcal{O}(NDC)$$

Proof : $\boldsymbol{Y}_{m,c} = \sum_{n=1}^{N}\left(\boldsymbol{L}_{m,n}\sum_{d=1}^{D}\boldsymbol{Q}_{m,d}\boldsymbol{K}_{n,d}\right)\boldsymbol{V}_{n,c}$

$$\boldsymbol{\mathcal{Z}}_{n,d,c} = \boldsymbol{K}_{n,d}\boldsymbol{V}_{n,c}$$

$$\boldsymbol{\mathcal{H}}_{m,d,c} = \sum_{n=1}^{N}\boldsymbol{L}_{m,n}\boldsymbol{\mathcal{Z}}_{n,d,c} = \sum_{n=1}^{N}\boldsymbol{L}_{m,n}\boldsymbol{K}_{n,d}\boldsymbol{V}_{n,c}$$

$$\boldsymbol{Y}_{m,c} = \sum_{d=1}^{D}\boldsymbol{Q}_{m,d}\boldsymbol{\mathcal{H}}_{m,d,c} = \sum_{d=1}^{D}\boldsymbol{Q}_{m,d}\left(\sum_{n=1}^{N}\boldsymbol{L}_{m,n}\boldsymbol{K}_{n,d}\boldsymbol{V}_{n,c}\right)$$

$$= \sum_{d=1}^{D}\sum_{n=1}^{N}\boldsymbol{Q}_{m,d}\boldsymbol{L}_{m,n}\boldsymbol{K}_{n,d}\boldsymbol{V}_{n,c}$$

$$= \sum_{n=1}^{N}\left(\sum_{d=1}^{D}\boldsymbol{Q}_{m,d}\boldsymbol{L}_{m,n}\boldsymbol{K}_{n,d}\right)\boldsymbol{V}_{n,c}$$

Corollary 1. *The complexity of* "$(\boldsymbol{Q}\boldsymbol{K}^{\top} \odot \boldsymbol{L})\boldsymbol{V}$" *with respect to N depends on* "$\text{einsum}[\text{MN}, \text{NDC} \rightarrow \text{MDC}](\boldsymbol{L}, \boldsymbol{\mathcal{Z}})$"

Theorem 1. *For any matrices* $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{N\times D}, \boldsymbol{L} \in \mathbb{R}^{N\times N}, \boldsymbol{V} \in \mathbb{R}^{N\times C}$*, the following equation holds:*

$$\boldsymbol{Y} = (\boldsymbol{Q}\boldsymbol{K}^{\top} \odot \boldsymbol{L})\boldsymbol{V} \quad \Longleftrightarrow \quad \begin{aligned} \mathcal{Z} &= \boldsymbol{K} \star \boldsymbol{V} = \text{einsum}[\text{ND}, \text{NC} \to \text{NDC}](\boldsymbol{K}, \boldsymbol{V}) && \mathcal{O}(NDC) \\ \mathcal{H} &= \text{einsum}[\text{MN}, \text{NDC} \to \text{MDC}](\boldsymbol{L}, \mathcal{Z}), \quad (\text{M} = \text{N}) && \mathcal{O}(N^2DC) \\ \boldsymbol{Y} &= \boldsymbol{Q} \star \mathcal{H} = \text{einsum}[\text{MD}, \text{MDC} \to \text{MC}](\boldsymbol{Q}, \mathcal{H}) && \mathcal{O}(NDC) \end{aligned}$$

Corollary 1. *The complexity of* "$(\boldsymbol{Q}\boldsymbol{K}^{\top} \odot \boldsymbol{L})\boldsymbol{V}$" *with respect to N depends on* "$\text{einsum}[\text{MN}, \text{NDC} \to \text{MDC}](\boldsymbol{L}, \mathcal{Z})$"

- Structured Masked Attention形式

$$\boldsymbol{Y} = (\boldsymbol{C}\boldsymbol{B}^{\top} \odot \boldsymbol{L})\boldsymbol{X} \quad \Longleftrightarrow \quad \begin{aligned} \mathcal{Z} &= \boldsymbol{B} \star \boldsymbol{X} = \text{einsum}[\text{ND}, \text{NC} \to \text{NDC}](\boldsymbol{B}, \boldsymbol{X}) \\ \mathcal{H} &= \text{einsum}[\text{MN}, \text{NDC} \to \text{MDC}](\boldsymbol{L}, \mathcal{Z}), \quad (\text{M} = \text{N}) \\ \boldsymbol{Y} &= \boldsymbol{C} \star \mathcal{H} = \text{einsum}[\text{MD}, \text{MDC} \to \text{MC}](\boldsymbol{C}, \mathcal{H}) \end{aligned}$$

$$\boldsymbol{H} = \text{reshape}\left[\mathbb{R}^{\text{N}\times\text{D}\times\text{C}} \to \mathbb{R}^{\text{N}\times\text{DC}}\right](\mathcal{H}), \quad \boldsymbol{Z} = \text{reshape}\left[\mathbb{R}^{\text{N}\times\text{D}\times\text{C}} \to \mathbb{R}^{\text{N}\times\text{DC}}\right](\mathcal{Z})$$

$$\mathcal{H} = \text{einsum}[\text{MN}, \text{NDC} \to \text{MDC}](\boldsymbol{L}, \mathcal{Z}) \quad \Longrightarrow \quad \boldsymbol{H} = \boldsymbol{L}\boldsymbol{Z} \quad \Longleftrightarrow \quad \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \\ \vdots \\ \boldsymbol{H}_{N-2} \\ \boldsymbol{H}_{N-1} \end{bmatrix} = \begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \\ \vdots \\ \boldsymbol{Z}_{N-2} \\ \boldsymbol{Z}_{N-1} \end{bmatrix}$$

Theorem 2. *For any 1-semiseparable structured matrix* $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ *and input tokens* $\boldsymbol{X} \in \mathbb{R}^{N\times C}$*, the calculation of* $\boldsymbol{Y} = \boldsymbol{L}\boldsymbol{X}$ *can be achieved with a complexity of* $\mathcal{O}(N)$ *by applying the* ***chunkwise algorithm***.

$$\underbrace{\begin{bmatrix} \boldsymbol{Y}_0 \\ \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_{N-2} \\ \boldsymbol{Y}_{N-1} \end{bmatrix}}_{\mathbb{R}^{N\times C}} = \underbrace{\begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix}}_{\mathbb{R}^{N\times N}} \underbrace{\begin{bmatrix} \boldsymbol{X}_0 \\ \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_{N-2} \\ \boldsymbol{X}_{N-1} \end{bmatrix}}_{\mathbb{R}^{N\times C}}$$

- Structured Masked Attention形式 *Complexity:* $\mathcal{O}(N^2) \longrightarrow \mathcal{O}(N)$

$$\begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \\ \vdots \\ \boldsymbol{H}_{N-2} \\ \boldsymbol{H}_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} A_{0:0} & 0 & \cdots & 0 & 0 \\ A_{1:0} & A_{1:1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \cdots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \cdots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix}}_{1-\textit{semiseparable structured matrix}} \begin{bmatrix} \boldsymbol{Z}_0 \\ \boldsymbol{Z}_1 \\ \vdots \\ \boldsymbol{Z}_{N-2} \\ \boldsymbol{Z}_{N-1} \end{bmatrix}$$

Chunkwise algorithm *For example, let sequence length $N = 3136$, chunk size $K = 56$*

$$\begin{bmatrix} \boldsymbol{Y}_0 \\ \boldsymbol{Y}_1 \\ \vdots \\ \boldsymbol{Y}_{N-2} \\ \boldsymbol{Y}_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} A_{0:0} & 0 & \dots & 0 & 0 \\ A_{1:0} & A_{1:1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N-2:0} & A_{N-2:1} & \dots & A_{N-2:N-2} & 0 \\ A_{N-1:0} & A_{N-1:1} & \dots & A_{N-1:N-2} & A_{N-1:N-1} \end{bmatrix}}_{1-\textit{semiseparable structured matrix}} \begin{bmatrix} \boldsymbol{X}_0 \\ \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_{N-2} \\ \boldsymbol{X}_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{54} \\ y_{55} \end{bmatrix} \\ \begin{bmatrix} y_{56} \\ y_{57} \\ \vdots \\ y_{110} \\ y_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} y_{3024} \\ y_{3025} \\ \vdots \\ y_{3078} \\ y_{3079} \end{bmatrix} \\ \begin{bmatrix} y_{3080} \\ y_{3081} \\ \vdots \\ y_{3134} \\ y_{3135} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & & & & \\ A_{1:0} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{54:0} & A_{54:1} & \cdots & 1 & \\ A_{55:0} & A_{55:1} & \cdots & A_{55:54} & 1 \end{bmatrix} & & & & \\ & \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \\ A_{57:0} & A_{57:1} & \cdots & A_{57:54} & A_{57:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{110:0} & A_{110:1} & \cdots & A_{110:54} & A_{110:55} \\ A_{111:0} & A_{111:1} & \cdots & A_{111:54} & A_{111:55} \end{bmatrix} & \begin{bmatrix} 1 & & & & \\ A_{57:56} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{110:56} & A_{110:57} & \cdots & 1 & \\ A_{111:56} & A_{111:57} & \cdots & A_{111:110} & 1 \end{bmatrix} & & \\ & \vdots & \vdots & & \ddots \\ & \begin{bmatrix} A_{3024:0} & A_{3024:1} & \cdots & A_{3024:54} & A_{3024:55} \\ A_{3025:0} & A_{3025:1} & \cdots & A_{3025:54} & A_{3025:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3078:0} & A_{3078:1} & \cdots & A_{3078:54} & A_{3078:55} \\ A_{3079:0} & A_{3079:1} & \cdots & A_{3079:54} & A_{3079:55} \end{bmatrix} & \begin{bmatrix} A_{3024:56} & A_{3024:57} & \cdots & A_{3024:110} & A_{3024:111} \\ A_{3025:56} & A_{3025:57} & \cdots & A_{3025:110} & A_{3025:111} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3078:56} & A_{3078:57} & \cdots & A_{3078:110} & A_{3078:111} \\ A_{3079:56} & A_{3079:57} & \cdots & A_{3079:110} & A_{3079:111} \end{bmatrix} & \cdots & \begin{bmatrix} 1 & & & & \\ A_{3025:3024} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3078:3024} & A_{3078:3025} & \cdots & 1 & \\ A_{3079:3024} & A_{3079:3025} & \cdots & A_{3079:3078} & 1 \end{bmatrix} & & \\ & \begin{bmatrix} A_{3080:0} & A_{3080:1} & \cdots & A_{3080:54} & A_{3080:55} \\ A_{3081:0} & A_{3081:1} & \cdots & A_{3081:54} & A_{3081:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:0} & A_{3134:1} & \cdots & A_{3134:54} & A_{3134:55} \\ A_{3135:0} & A_{3135:1} & \cdots & A_{3135:54} & A_{3135:55} \end{bmatrix} & \begin{bmatrix} A_{3080:56} & A_{3080:57} & \cdots & A_{3080:110} & A_{3080:111} \\ A_{3081:56} & A_{3081:57} & \cdots & A_{3081:110} & A_{3081:111} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:56} & A_{3134:57} & \cdots & A_{3134:110} & A_{3134:111} \\ A_{3135:56} & A_{3135:57} & \cdots & A_{3135:110} & A_{3135:111} \end{bmatrix} & \cdots & \begin{bmatrix} A_{3080:3024} & A_{3080:3025} & \cdots & A_{3080:3078} & A_{3080:3079} \\ A_{3081:3024} & A_{3081:3025} & \cdots & A_{3081:3078} & A_{3081:3079} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:3024} & A_{3134:3025} & \cdots & A_{3134:3078} & A_{3134:3079} \\ A_{3135:3024} & A_{3135:3025} & \cdots & A_{3135:3078} & A_{3135:3079} \end{bmatrix} & \begin{bmatrix} 1 & & & & \\ A_{3081:3080} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3134:3080} & A_{3134:3081} & \cdots & 1 & \\ A_{3135:3080} & A_{3135:3081} & \cdots & A_{3135:3134} & 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix} \\ \begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix} \\ \begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix} \end{bmatrix}$$

$$
\begin{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{54} \\ y_{55} \end{bmatrix} \\ \begin{bmatrix} y_{56} \\ y_{57} \\ \vdots \\ y_{110} \\ y_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} y_{3024} \\ y_{3025} \\ \vdots \\ y_{3078} \\ y_{3079} \end{bmatrix} \\ \begin{bmatrix} y_{3080} \\ y_{3081} \\ \vdots \\ y_{3134} \\ y_{3135} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & & & & \\ A_{1:0} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{54:0} & A_{54:1} & \cdots & 1 & \\ A_{55:0} & A_{55:1} & \cdots & A_{55:54} & 1 \end{bmatrix} & & & & \\ & \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \\ A_{57:0} & A_{57:1} & \cdots & A_{57:54} & A_{57:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{110:0} & A_{110:1} & \cdots & A_{110:54} & A_{110:55} \\ A_{111:0} & A_{111:1} & \cdots & A_{111:54} & A_{111:55} \end{bmatrix} & \begin{bmatrix} 1 & & & & \\ A_{57:56} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{110:56} & A_{110:57} & \cdots & 1 & \\ A_{111:56} & A_{111:57} & \cdots & A_{111:110} & 1 \end{bmatrix} & & \\ & \vdots & \vdots & \ddots & \\ \begin{bmatrix} A_{3024:0} & A_{3024:1} & \cdots & A_{3024:54} & A_{3024:55} \\ A_{3025:0} & A_{3025:1} & \cdots & A_{3025:54} & A_{3025:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3078:0} & A_{3078:1} & \cdots & A_{3078:54} & A_{3078:55} \\ A_{3079:0} & A_{3079:1} & \cdots & A_{3079:54} & A_{3079:55} \end{bmatrix} & \begin{bmatrix} A_{3024:56} & A_{3024:57} & \cdots & A_{3024:110} & A_{3024:111} \\ A_{3025:56} & A_{3025:57} & \cdots & A_{3025:110} & A_{3025:111} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3078:56} & A_{3078:57} & \cdots & A_{3078:110} & A_{3078:111} \\ A_{3079:56} & A_{3079:57} & \cdots & A_{3079:110} & A_{3079:111} \end{bmatrix} & \cdots & \begin{bmatrix} 1 & & & & \\ A_{3025:3024} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3078:3024} & A_{3078:3025} & \cdots & 1 & \\ A_{3079:3024} & A_{3079:3025} & \cdots & A_{3079:3078} & 1 \end{bmatrix} & \\ \begin{bmatrix} A_{3080:0} & A_{3080:1} & \cdots & A_{3080:54} & A_{3080:55} \\ A_{3081:0} & A_{3081:1} & \cdots & A_{3081:54} & A_{3081:55} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:0} & A_{3134:1} & \cdots & A_{3134:54} & A_{3134:55} \\ A_{3135:0} & A_{3135:1} & \cdots & A_{3135:54} & A_{3135:55} \end{bmatrix} & \begin{bmatrix} A_{3080:56} & A_{3080:57} & \cdots & A_{3080:110} & A_{3080:111} \\ A_{3081:56} & A_{3081:57} & \cdots & A_{3081:110} & A_{3081:111} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:56} & A_{3134:57} & \cdots & A_{3134:110} & A_{3134:111} \\ A_{3135:56} & A_{3135:57} & \cdots & A_{3135:110} & A_{3135:111} \end{bmatrix} & \cdots & \begin{bmatrix} A_{3080:3024} & A_{3080:3025} & \cdots & A_{3080:3078} & A_{3080:3079} \\ A_{3081:3024} & A_{3081:3025} & \cdots & A_{3081:3078} & A_{3081:3079} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3134:3024} & A_{3134:3025} & \cdots & A_{3134:3078} & A_{3134:3079} \\ A_{3135:3024} & A_{3135:3025} & \cdots & A_{3135:3078} & A_{3135:3079} \end{bmatrix} & \begin{bmatrix} 1 & & & \\ A_{3081:3080} & 1 & & \\ \vdots & \vdots & \ddots & \\ A_{3134:3080} & A_{3134:3081} & \cdots & 1 \\ A_{3135:3080} & A_{3135:3081} & \cdots & A_{3135:3134} \quad 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix} \\ \begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix} \\ \begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix} \end{bmatrix}
$$

$$
\begin{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{54} \\ y_{55} \end{bmatrix} \\ \begin{bmatrix} y_{56} \\ y_{57} \\ \vdots \\ y_{110} \\ y_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} y_{3024} \\ y_{3025} \\ \vdots \\ y_{3078} \\ y_{3079} \end{bmatrix} \\ \begin{bmatrix} y_{3080} \\ y_{3081} \\ \vdots \\ y_{3134} \\ y_{3135} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & & & & \\ A_{1:0} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{54:0} & A_{54:1} & \cdots & 1 & \\ A_{55:0} & A_{55:1} & \cdots & A_{55:54} & 1 \end{bmatrix} & & & & \\ \begin{bmatrix} 1 \\ A_{57:56} \\ \vdots \\ A_{110:56} \\ A_{111:56} \end{bmatrix} A_{56:56}[A_{56:0} \quad A_{56:1} \quad \cdots \quad A_{56:54} \quad A_{56:55}] & \begin{bmatrix} 1 & & & & \\ A_{57:56} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{110:56} & A_{110:57} & \cdots & 1 & \\ A_{111:56} & A_{111:57} & \cdots & A_{111:110} & 1 \end{bmatrix} & & \\ & \vdots & \vdots & \ddots & \\ \begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:56}[A_{56:0} \quad A_{56:1} \quad \cdots \quad A_{56:54} \quad A_{56:55}] & \begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:112}[A_{112:56} \quad A_{112:57} \quad \cdots \quad A_{112:110} \quad A_{112:111}] & \cdots & \begin{bmatrix} 1 & & & & \\ A_{3025:3024} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3078:3024} & A_{3078:3025} & \cdots & 1 & \\ A_{3079:3024} & A_{3079:3025} & \cdots & A_{3079:3078} & 1 \end{bmatrix} & \\ \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:56}[A_{56:0} \quad A_{56:1} \quad \cdots \quad A_{56:54} \quad A_{56:55}] & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:112}[A_{112:56} \quad A_{112:57} \quad \cdots \quad A_{112:110} \quad A_{112:111}] & \cdots & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:3080}[A_{3080:3024} \quad A_{3080:3025} \quad \cdots \quad A_{3080:3078} \quad A_{3080:3079}] & \begin{bmatrix} 1 & & & \\ A_{3081:3080} & 1 & & \\ \vdots & \vdots & \ddots & \\ A_{3134:3080} & A_{3134:3081} & \cdots & 1 \\ A_{3135:3080} & A_{3135:3081} & \cdots & A_{3135:3134} \quad 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix} \\ \begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix} \\ \begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix} \end{bmatrix}
$$

$$
\boldsymbol{Y}_{diag}
$$

Complexity: $\mathcal{O}(\frac{N}{K} \times K^2)$

$$
Y^{diag} = \begin{bmatrix}
\begin{bmatrix} 1 & & & & \\ A_{1:0} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{54:0} & A_{54:1} & \cdots & 1 & \\ A_{55:0} & A_{55:1} & \cdots & A_{55:54} & 1 \end{bmatrix} & & & & \\
& \begin{bmatrix} 1 & & & & \\ A_{57:56} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{110:56} & A_{110:57} & \cdots & 1 & \\ A_{111:56} & A_{111:57} & \cdots & A_{111:110} & 1 \end{bmatrix} & & & \\
& \vdots & \ddots & & \\
& & & \begin{bmatrix} 1 & & & & \\ A_{3025:3024} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3078:3024} & A_{3078:3025} & \cdots & 1 & \\ A_{3079:3024} & A_{3079:3025} & \cdots & A_{3079:3078} & 1 \end{bmatrix} & \\
& & & & \begin{bmatrix} 1 & & & & \\ A_{3081:3080} & 1 & & & \\ \vdots & \vdots & \ddots & & \\ A_{3134:3080} & A_{3134:3081} & \cdots & 1 & \\ A_{3135:3080} & A_{3135:3081} & \cdots & A_{3135:3134} & 1 \end{bmatrix}
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix}
\begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix}
\vdots
\begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix}
\begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix}
$$

$$
Y^{tril} = \begin{bmatrix}
\begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & & & & \\
\begin{bmatrix} 1 \\ A_{57:56} \\ \vdots \\ A_{110:56} \\ A_{111:56} \end{bmatrix} A_{56:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & & & \\
\vdots & \vdots & \ddots & & \\
\begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:112} \begin{bmatrix} A_{112:56} & A_{112:57} & \cdots & A_{112:110} & A_{112:111} \end{bmatrix} & \cdots & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & \\
\begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:112} \begin{bmatrix} A_{112:56} & A_{112:57} & \cdots & A_{112:110} & A_{112:111} \end{bmatrix} & \cdots & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:3080} \begin{bmatrix} A_{3080:3024} & A_{3080:3025} & \cdots & A_{3080:3078} & A_{3080:3079} \end{bmatrix} & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix}
\begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix}
\vdots
\begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix}
\begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix}
$$

$$
Y^{tril} = \begin{bmatrix}
\begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & & & \\
\begin{bmatrix} 1 \\ A_{57:56} \\ \vdots \\ A_{110:56} \\ A_{111:56} \end{bmatrix} A_{56:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & & \\
\vdots & \vdots & \ddots & \\
\begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:112} \begin{bmatrix} A_{112:56} & A_{112:57} & \cdots & A_{112:110} & A_{112:111} \end{bmatrix} & \cdots & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:56} \begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:112} \begin{bmatrix} A_{112:56} & A_{112:57} & \cdots & A_{112:110} & A_{112:111} \end{bmatrix} & \cdots & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:3080} \begin{bmatrix} A_{3080:3024} & A_{3080:3025} & \cdots & A_{3080:3078} & A_{3080:3079} \end{bmatrix} & \begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & \cdots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}
\end{bmatrix}
\begin{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix} \\ \begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix} \\ \begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix} \end{bmatrix}
$$

$$
\begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{54} \\ s_{55} \end{bmatrix} = \begin{bmatrix}
\begin{bmatrix} A_{56:0} & A_{56:1} & \cdots & A_{56:54} & A_{56:55} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{54} \\ x_{55} \end{bmatrix} \\
\begin{bmatrix} A_{112:56} & A_{112:57} & \cdots & A_{112:110} & A_{112:111} \end{bmatrix} \begin{bmatrix} x_{56} \\ x_{57} \\ \vdots \\ x_{110} \\ x_{111} \end{bmatrix} \\
\vdots \\
\begin{bmatrix} A_{3080:3024} & A_{3080:3025} & \cdots & A_{3080:3078} & A_{3080:3079} \end{bmatrix} \begin{bmatrix} x_{3024} \\ x_{3025} \\ \vdots \\ x_{3078} \\ x_{3079} \end{bmatrix} \\
\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{3080} \\ x_{3081} \\ \vdots \\ x_{3134} \\ x_{3135} \end{bmatrix}
\end{bmatrix},
$$

Complexity: $\mathcal{O}(\frac{N}{K} \times K)$

$$
Y^{tril} = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 \\
\begin{bmatrix} 1 \\ A_{57:56} \\ \vdots \\ A_{110:56} \\ A_{111:56} \end{bmatrix} A_{56:56} & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:56} & \begin{bmatrix} 1 \\ A_{3025:3024} \\ \vdots \\ A_{3078:3024} \\ A_{3079:3024} \end{bmatrix} A_{3024:112} & \cdots & 0 & 0 \\
\begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:56} & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:112} & \cdots & \begin{bmatrix} 1 \\ A_{3081:3080} \\ \vdots \\ A_{3134:3080} \\ A_{3135:3080} \end{bmatrix} A_{3080:3080} & 0
\end{bmatrix}
\begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{54} \\ s_{55} \end{bmatrix} = \begin{bmatrix}
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
\begin{bmatrix} C_{56}1 \\ C_{57}A_{57:56} \\ \vdots \\ C_{110}A_{110:56} \\ C_{111}A_{111:56} \end{bmatrix} \\
\vdots \\
\begin{bmatrix} C_{2024} \\ C_{2025}A_{3025:3024} \\ \vdots \\ C_{3078}A_{3078:3024} \\ C_{3079}A_{3079:3024} \end{bmatrix} \\
\begin{bmatrix} C_{3080} \\ C_{3081}A_{3081:3080} \\ \vdots \\ C_{3134}A_{3134:3080} \\ C_{3135}A_{3135:3080} \end{bmatrix}
\end{bmatrix}
\underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ A_{56:56} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{3024:56} & A_{3024:112} & \cdots & 0 & 0 \\ A_{3080:56} & A_{3080:112} & \cdots & A_{3080:3080} & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{54} \\ s_{55} \end{bmatrix}}_{Loop}
$$

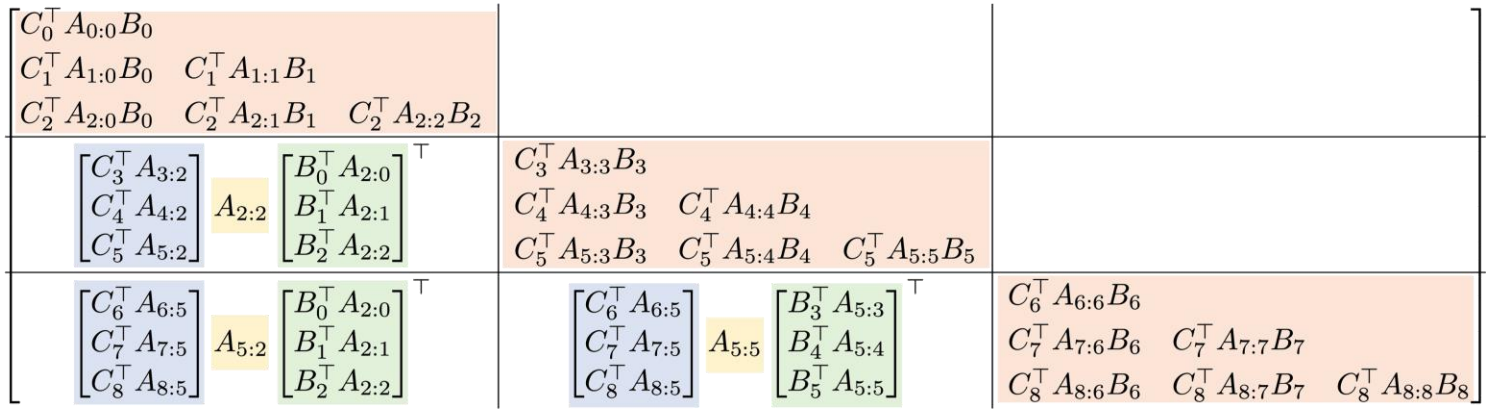
Complexity: $\mathcal{O}(\frac{N}{K} + N)$

4.3 Mamba2: The Algorithm

- Mamba2: Structured Masked Attention

$$\boldsymbol{Y} = (\boldsymbol{C}\boldsymbol{B}^\top \odot \boldsymbol{L})\boldsymbol{X} = \boldsymbol{C} \star \big(\boldsymbol{L} \times (\boldsymbol{B} \star \boldsymbol{X})\big)$$

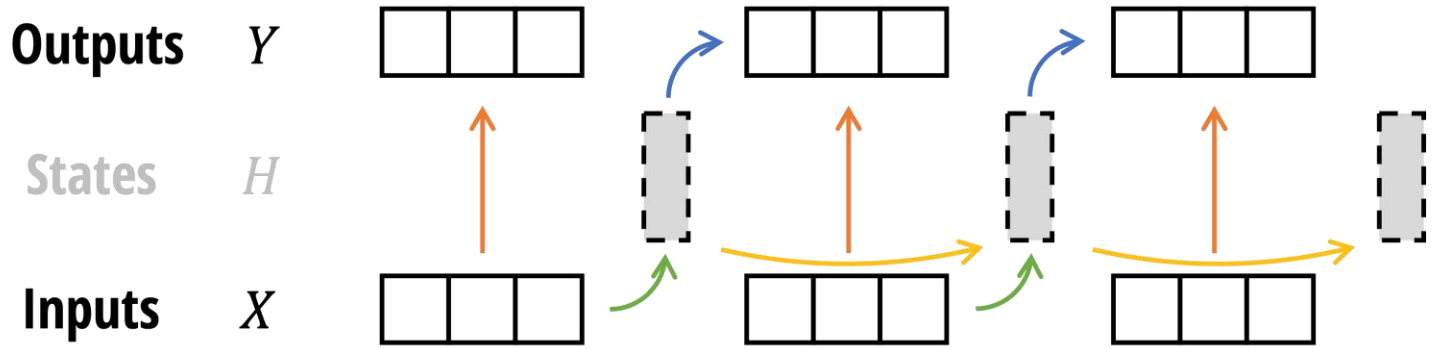
Complexity: $\mathcal{O}(N^2) \longrightarrow \mathcal{O}(N)$



Semiseparable Matrix M

Block Decomposition

- Diagonal Block: Input → Output
- Low-Rank Block: Input → State
- Low-Rank Block: State → State
- Low-Rank Block: State → Output



4.4 Mamba2: The Systems

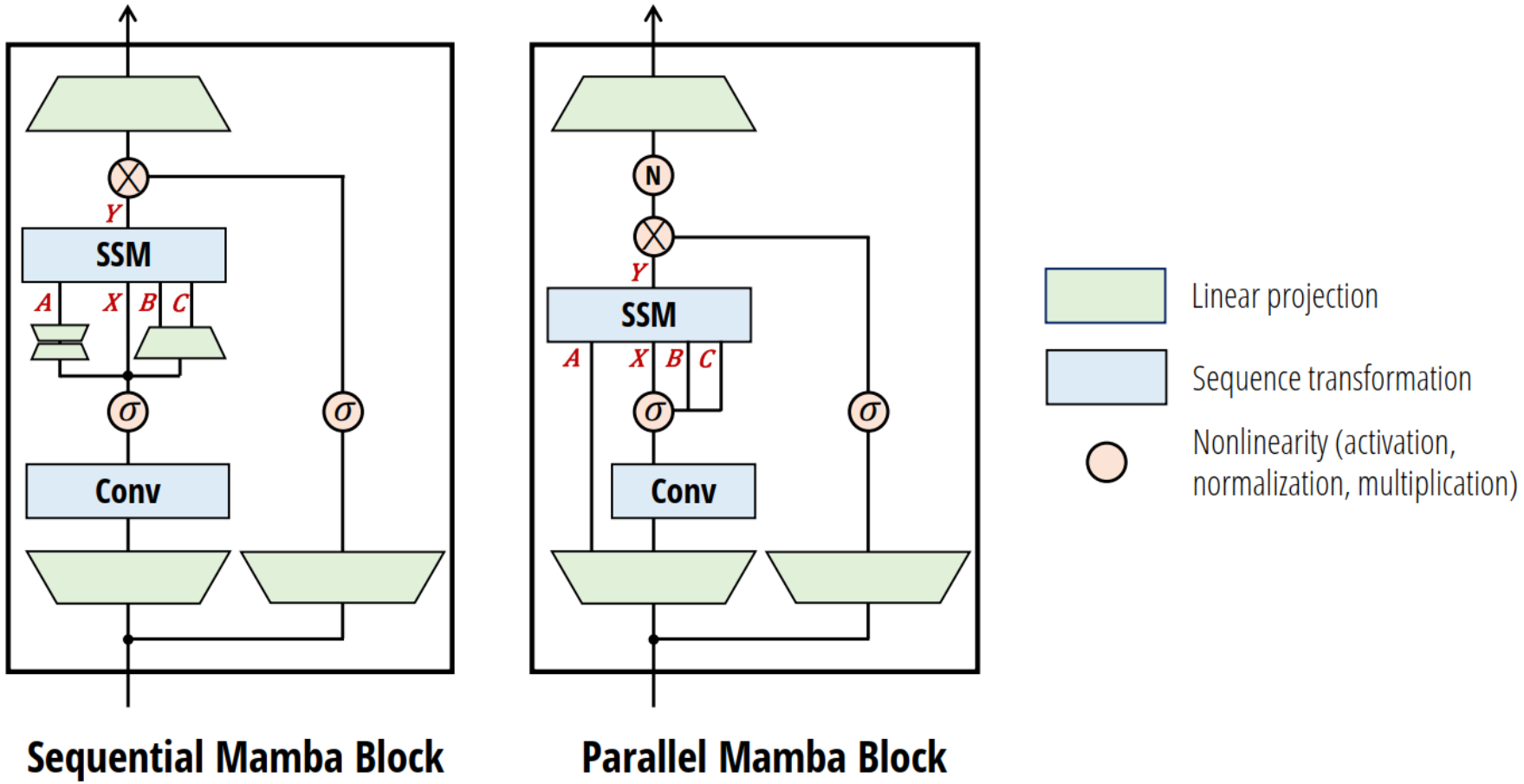


Figure 6: (**Mamba-2 Architecture**.) The Mamba-2 block simplifies the Mamba block by removing sequential linear projections; the SSM parameters A, B, C are produced at the beginning of the block instead of as a function of the SSM input X. An additional normalization layer is added as in NormFormer (Shleifer, Weston, and Ott 2021), improving stability. The B and C projections only have a single head shared across the X heads, analogous to multi-value attention (MVA).

4.4 Mamba2: The Systems

	Self-Attention	Mamba (SSM)	**Mamba2 (SSD)**
State size	N	1	1
Training FLOPs	N^2	N	N
Inference FLOPs	N	1	1
(Naive) memory	N^2	N	N
implementation	CUDA	CUDA	Triton
Matrix multiplication	✅	❌	✅

	PyTorch	**CUDA kernel**	**Triton kernel**
性能表现	🟡 中等，通用优化	🟢🟢 最高，可精细优化	🟢 高，结构友好自动并行优化
易用性	🟢 非常高，调用即用	🔴 最低，需熟悉 CUDA 编程	🟡 需要写 kernel
灵活性	🟡 中等	🟢🟢 最高，完全掌控硬件细节	🟢 高，支持结构感知优化
编译时间	🟢 无编译，直接运行	🔴 静态编译，耗时且复杂	🟡 动态 JIT 编译
适合场景	通用矩阵/张量操作	极限性能场景、重复运行模型	Transformer, SSM等结构优化

4.4 Mamba2: The Systems

- Experiment results
  - Mamba-2 (SSD) is significantly **faster** and scales much better than Mamba-1(Scan)

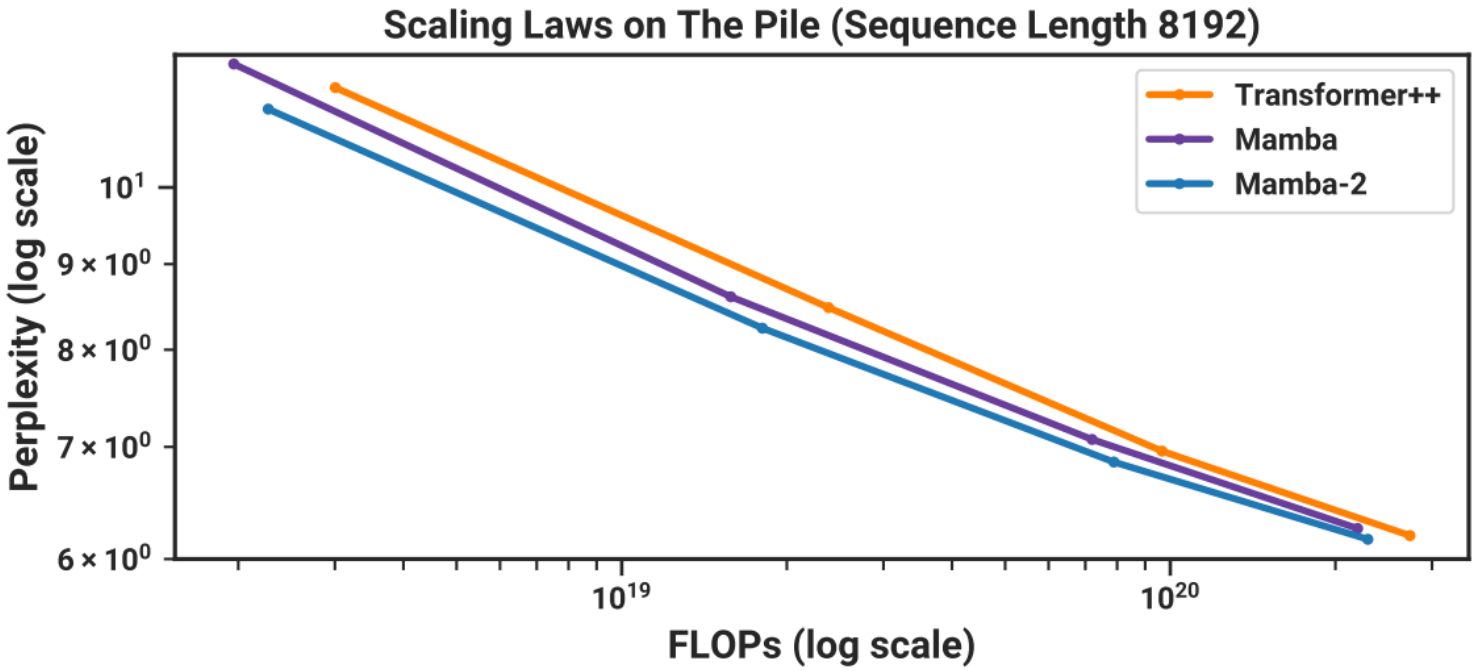


Figure 9: (**Scaling Laws.**) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba-2 matches or exceeds the performance of Mamba as well as a strong "Transformer++" recipe. Compared to our Transformer baseline, Mamba-2 is Pareto dominant on performance (perplexity), theoretical FLOPs, and actual wall-clock time.

4.4 Mamba2: The Systems

- Experiment results
  - Mamba-2 (SSD) is significantly **faster** and scales much better than Mamba-1(Scan)

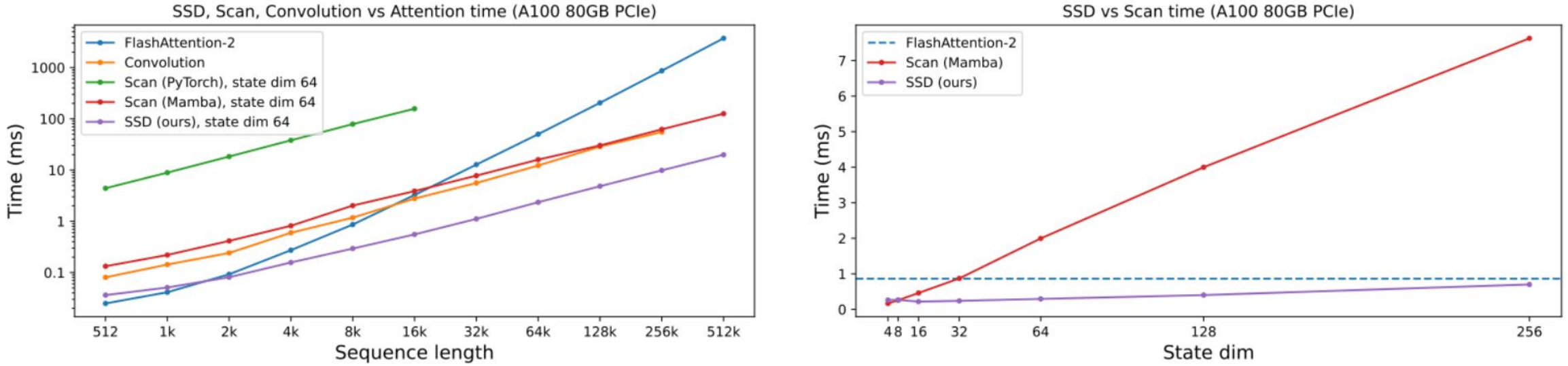


Figure 10: (**Efficiency Benchmarks.**) (*Left*) Our SSD is 2 – 8× faster than a Mamba fused scan for large state expansion ($N = 64$) and faster than FlashAttention-2 for sequence length 2k and above. (*Right*) Sequence length 4K: Increasing state expansion slows down the Mamba optimized scan implementation linearly. SSD can handle much larger state expansion factors without much slowdown.

目录

5.1 PPMA: Motivation (Current Issue)

Question: Have SOTA Mamba-based models outperform SOTA ViTs in computer vision domain, especially on high-level vision tasks?

Not yet, experimental results speak louder than words

Table 5: Image classification performance on the ImageNet-1K.

Model	Arch.	#Param. (M)	FLOPs (G)	Top-1 (%)
ConvNeXt-T [32]	CNN	29	4.5	82.1
EffNet-B4 [41]		19	4.2	82.9
VMamba-T [30]	SSM	30	4.9	82.6
GrootVL-T [48]		30	4.8	83.4
Spatial-Mamba-T [46]		27	4.5	83.5
MLLA-T [15]		25	4.2	83.5
Swin-T [31]	Trans.	29	4.5	82.1
NAT-T [16]		28	4.3	83.2
BiFormer-S [56]		26	4.5	83.8
RMT-S [10]		27	4.5	84.0
ConvNeXt-S [32]	CNN	50	8.7	83.1
EffNet-B5 [41]		30	9.9	83.6
VMamba-S [30]	SSM	50	8.7	83.6
GrootVL-S [48]		51	8.5	84.2
MLLA-S [15]		43	7.3	84.4
Spatial-Mamba-S [46]		43	7.1	84.6
Swin-S [31]	Trans.	50	8.7	83.0
NAT-S [16]		51	7.8	83.7
BiFormer-B [56]		57	9.8	84.3
iFormer-B [37]		48	9.4	84.6
RMT-B [10]		54	9.7	84.9

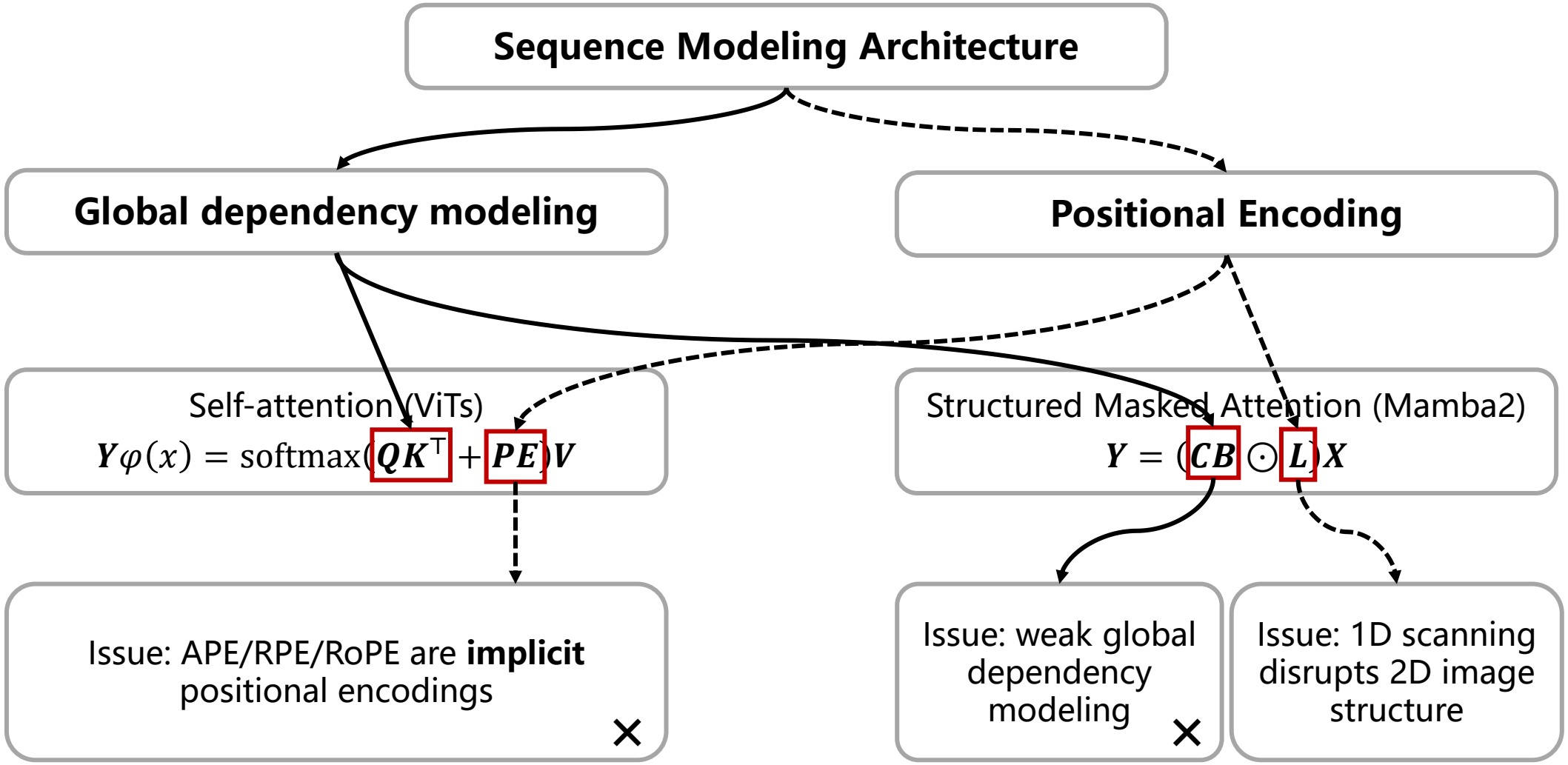
Table 6: Object detection and instance segmentation performance on COCO.

Backbone	Arch.	#Param. (M)	FLOPs (G)	AP^b	AP^m
ResNet-50 [19]	CNN	44	260	38.2	34.7
ConvNeXt-T [32]		48	262	44.2	40.1
MLLA-T [15]	SSM	44	255	46.8	42.1
GrootVL-T [48]		49	265	47.0	42.7
VMamba-T [30]		50	271	47.3	42.7
Spatial-Mamba-T [46]		46	216	47.6	42.9
Swin-T [31]	Trans.	48	267	43.7	39.8
CSWin-T [8]		42	279	46.7	42.2
BiFormer-S [56]		–	–	47.8	43.2
RMT-S [10]		46	262	48.8	43.6
ResNet-101 [19]	CNN	63	336	40.4	36.4
ConvNeXt-S [32]		70	348	45.4	41.8
GrootVL-S [48]	SSM	70	341	48.6	43.6
VMamba-S [30]		70	349	48.7	43.7
Spatial-Mamba-S [46]		63	315	49.2	44.0
MLLA-S [15]		63	319	49.2	44.2
Swin-S [31]	Trans.	69	359	45.7	41.1
CSWin-S [8]		54	342	47.9	43.2
BiFormer-B [56]		–	–	48.6	43.7
RMT-B [10]		73	373	50.7	45.1

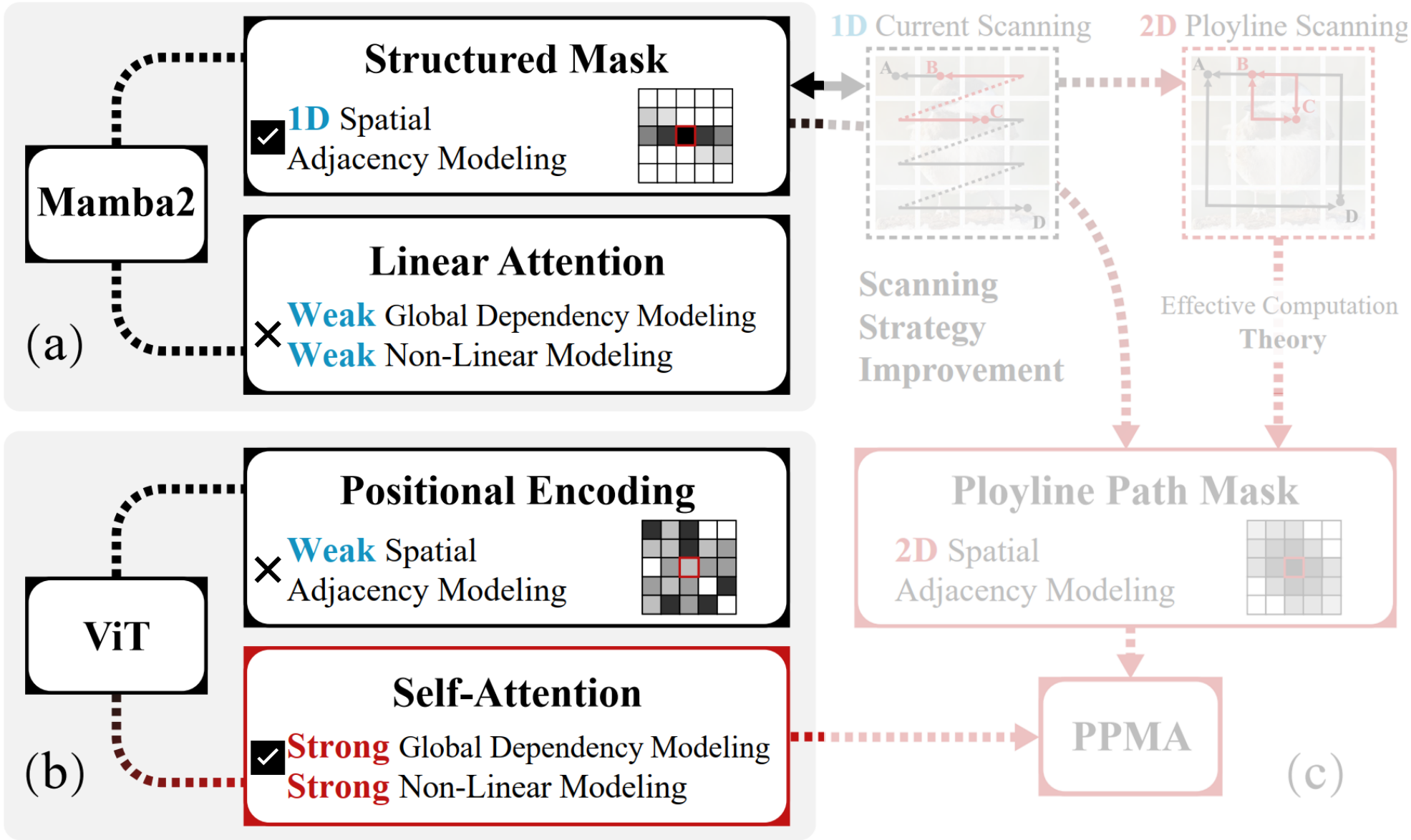
Table 7: Semantic segmentation performance on ADE20K.

Backbone	Arch.	#Param. (M)	FLOPs (G)	mIoU(%) SS	mIoU(%) MS
ResNet-50 [19]	CNN	67	953	42.1	42.8
ConvNeXt-T [32]		60	939	46.0	46.7
VMamba-T [30]	SSM	62	949	48.0	48.8
2DMamba-T [52]		62	950	48.6	49.3
GrootVL-T [48]		60	941	48.5	49.4
Spatial-Mamba-S [46]		57	936	48.6	49.4
Swin-T [31]	Trans.	60	945	44.4	45.8
NAT-T [16]		58	934	47.1	48.4
BiFormer-S [56]		–	–	49.8	50.8
RMT-S [10]		56	937	49.8	49.7
ResNet-101 [19]	CNN	85	1030	42.9	44.0
ConvNeXt-S [32]		82	1027	48.7	49.6
VMamba-S [30]	SSM	82	1028	50.6	51.2
Spatial-Mamba-S [46]		73	992	50.6	51.4
GrootVL-S [48]		82	1019	50.7	51.7
Swin-S [31]	Trans.	81	1039	47.6	49.5
NAT-S [16]		82	1010	48.0	49.5
BiFormer-B [56]		–	–	51.0	51.7
RMT-B [10]		83	1051	52.0	52.1

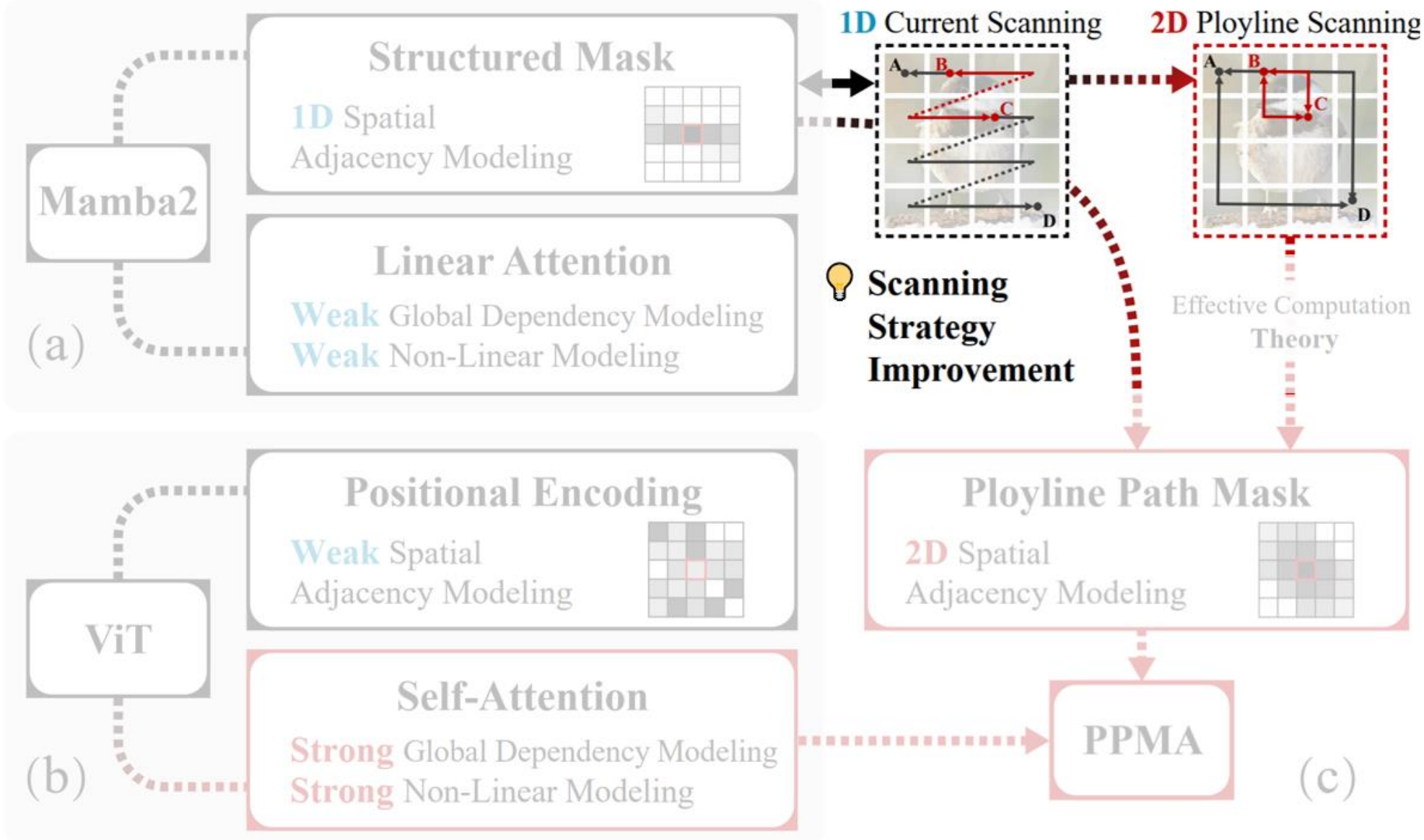
5.1 PPMA: Motivation (Current Issue)



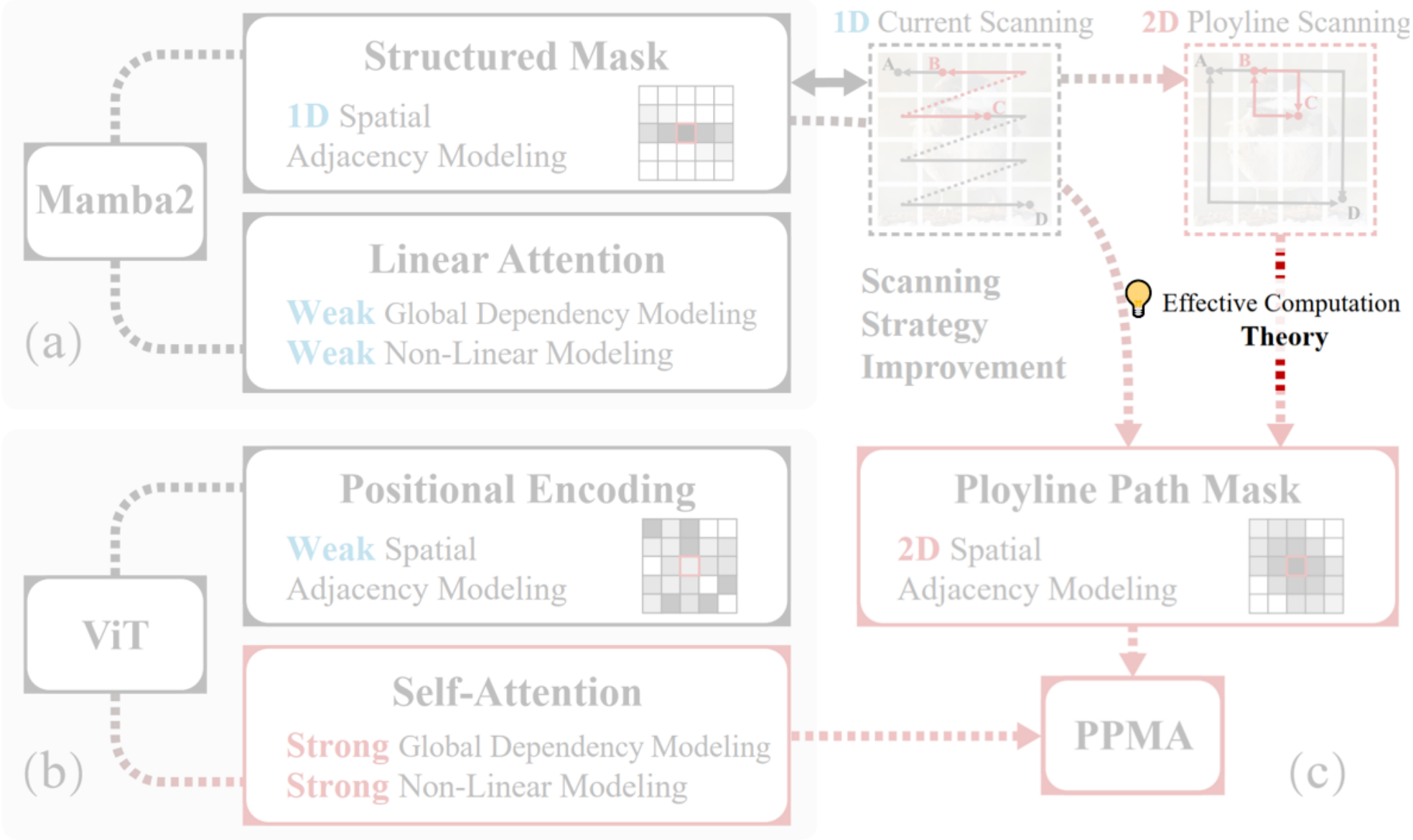
*We adapt Mamba2's 1D structured mask to **2D polyline path mask** and integrate it into the self-attention mechanism of ViTs as an **explicit** positional encoding.*



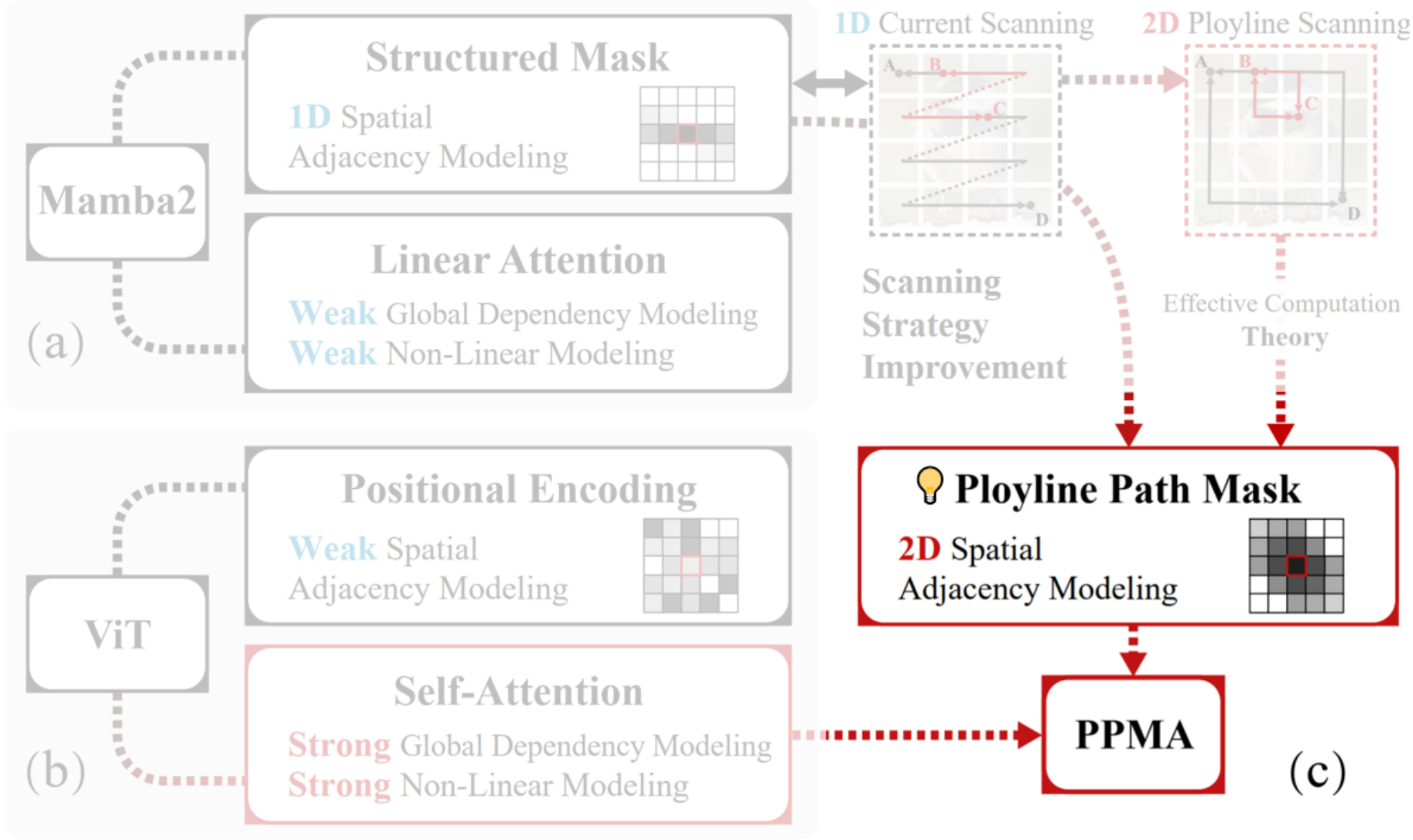
*We adapt Mamba2's 1D structured mask to **2D polyline path mask** and integrate it into the self-attention mechanism of ViTs as an **explicit** positional encoding.*



*We adapt Mamba2's 1D structured mask to **2D polyline path mask** and integrate it into the self-attention mechanism of ViTs as an **explicit** positional encoding.*



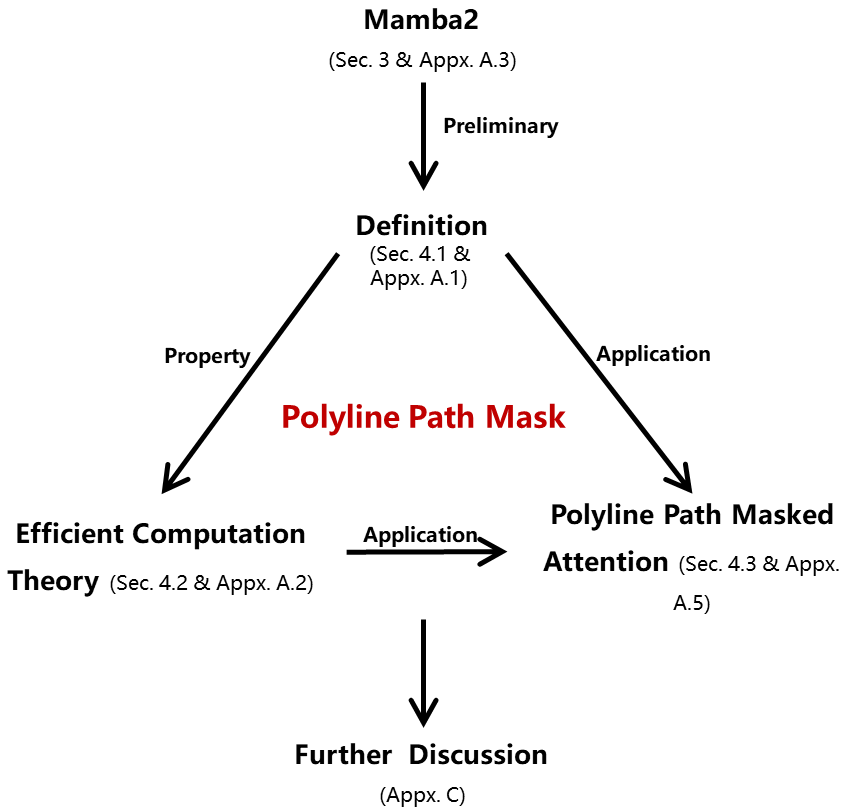
*We adapt Mamba2's 1D structured mask to **2D polyline path mask** and integrate it into the self-attention mechanism of ViTs as an **explicit** positional encoding.*



5.1 PPMA: Insight & Contribution

- Our insight: Mamba2 core mechanism is **structured mask**
  - **Explicit positional encoding** through the recursive propagation mechanism
  - **Semantic continuity awareness** in sequences through the **selective** mechanism

- Our contribution
  - A novel **2D polyline path structured mask**
  - An **efficient algorithm** for the calculation of the polyline path mask
  - **Polyline Path Masked (Sparse) Attention**
  - **SOTA performance** on image classification, object detection, and segmentation tasks compared to SSM-based models and ViTs



5.2 PPMA: Method (Definition of Polyline Path Mask)

- 2D polyline path scanning
  - Current issue: fail to preserve the distance relationship of 2D tokens
  - Our solution: scan the 2D tokens along multiple paths

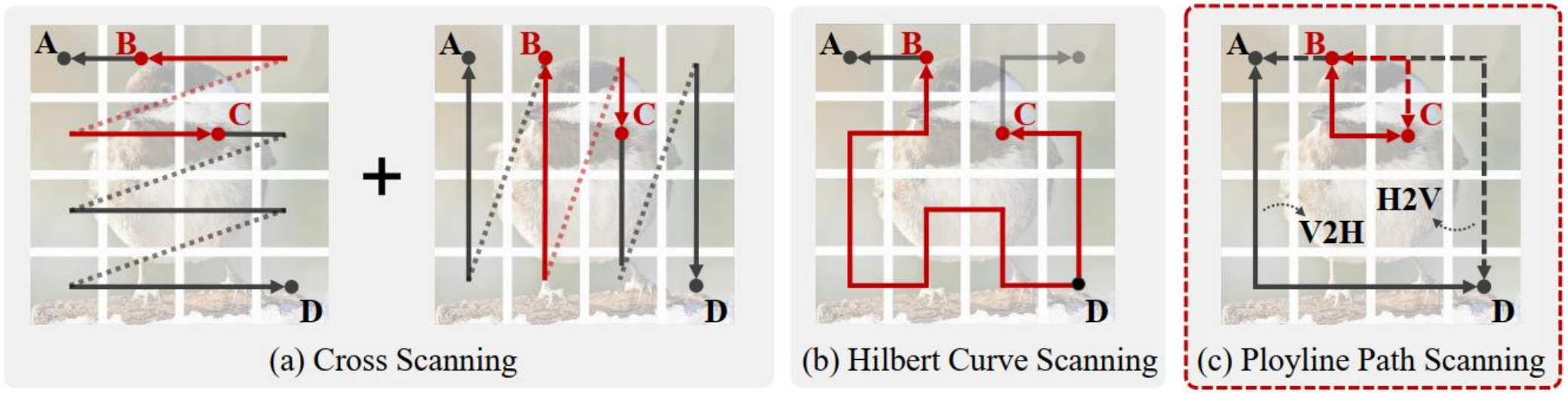


Figure 2: Compared to existing scanning strategies (a) and (b), which flatten 2D tokens into a 1D sequence, our polyline path scanning (c) better preserves the adjacency of 2D tokens.

5.2 PPMA: Method (Definition of Polyline Path Mask)

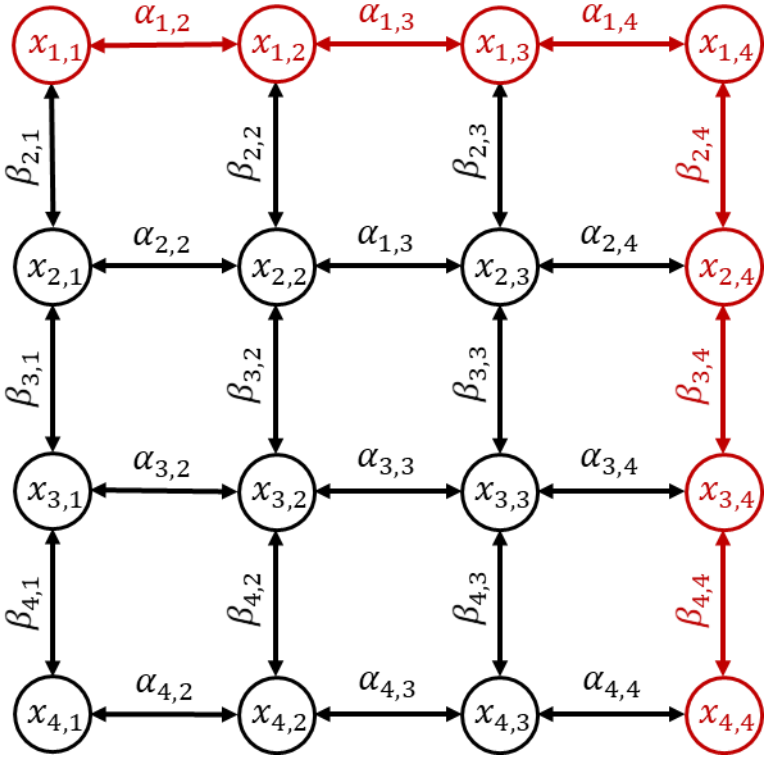
- 2D polyline path mask
  - Learn the **horizontal** factor and **vertical** decay factors

$$\begin{cases} \alpha_{i,j} = \exp(-\text{Softplus}(\text{MLP}_{\alpha}(\boldsymbol{x}_{i,j}))) \in \mathbb{R}^1 \ (\mathbf{0}\sim\mathbf{1}) \\ \beta_{i,j} = \exp(-\text{Softplus}(\text{MLP}_{\alpha}(\boldsymbol{x}_{i,j}))) \in \mathbb{R}^1 \ (\mathbf{0}\sim\mathbf{1}) \end{cases}$$

  - Calculate the decay weight of each polyline path, i.e., for path from $\boldsymbol{x}_{k,l}$ to $\boldsymbol{x}_{i,j}$:

$$\mathcal{L}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$$

$$\alpha_{i,j:l} = \begin{cases} \alpha_{i,j+1} \times \cdots \times \alpha_{i,l} & if\ j < l \\ 1 & if\ j = l \\ \alpha_{i,l+1} \times \cdots \times \alpha_{i,j} & if\ j > l \end{cases}, \quad \beta_{i:k,l} = \begin{cases} \beta_{i+1,l} \times \cdots \times \beta_{k,l} & if\ i < k \\ 1 & if\ i = k \\ \beta_{k+1,l} \times \cdots \times \beta_{i,l} & if\ i > k \end{cases}$$



$$\mathcal{L}_{1,1,4,4} = \alpha_{1,2}\alpha_{1,3}\alpha_{1,4}\beta_{2,4}\beta_{3,4}\ \beta_{4,4}$$

5.2 PPMA: Method (Definition of Polyline Path Mask)

- 2D polyline path mask
  - Learn the horizontal factor and vertical decay factors
  - Calculate the decay weight of each polyline path, i.e., for path from $\boldsymbol{x}_{k,l}$ to $\boldsymbol{x}_{i,j}$:

$$\boldsymbol{\mathcal{L}}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$$

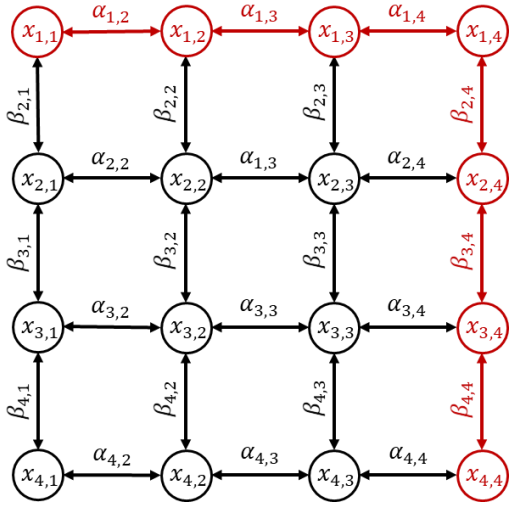
  - Combine bidirectional vertical-then-horizontal path and horizontal-then-vertical path:

$$\boldsymbol{\mathcal{L}}^{2D} = \boldsymbol{\mathcal{L}} + \widetilde{\boldsymbol{\mathcal{L}}}, \quad \widetilde{\boldsymbol{\mathcal{L}}}_{i,j,k,l} = \alpha_{k,j:l}\beta_{i:k,j} = \boldsymbol{\mathcal{L}}_{k,l,i,j}$$

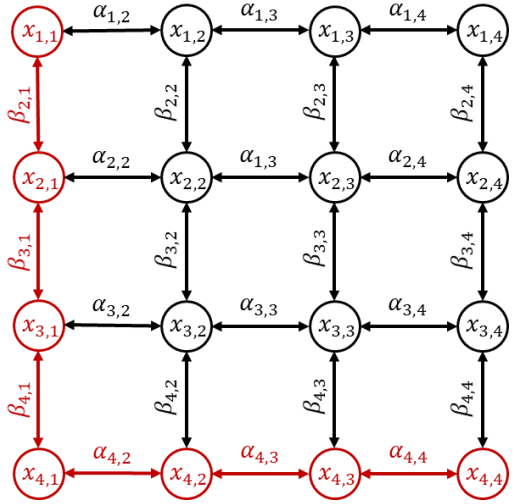
  - Unfold 4D tensors $\boldsymbol{\mathcal{L}}^{2D} \in \mathbb{R}^{H\times W\times H\times W}$ to 2D matrix $\boldsymbol{L}^{2D} \in \mathbb{R}^{HW\times HW}$:

$$\boldsymbol{L}^{2D} = \text{unfold}(\boldsymbol{\mathcal{L}}^{2D}), \quad \boldsymbol{L}^{2D}_{(i-1)\times W+j,(k-1)\times W+l,} = \boldsymbol{\mathcal{L}}^{2D}_{i,j,k,l}$$

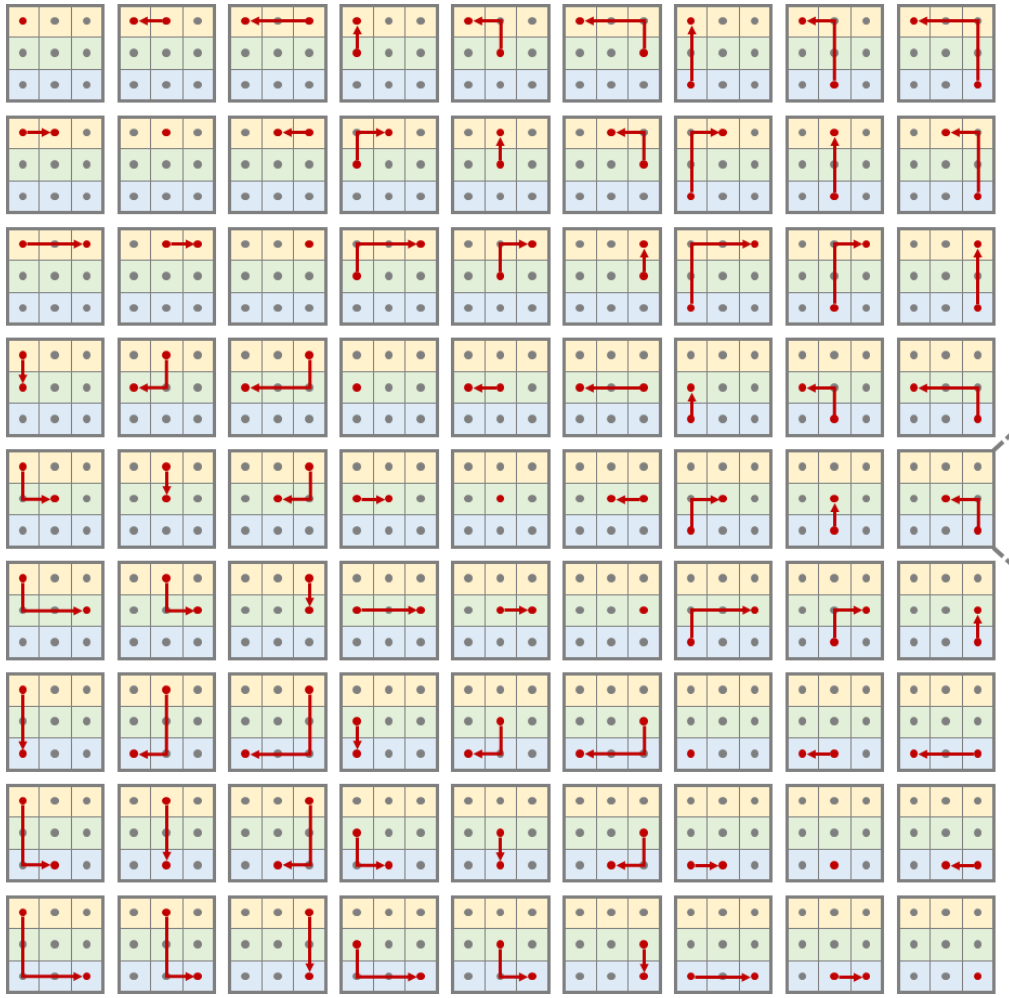
$$\boldsymbol{L} = \text{unfold}(\boldsymbol{\mathcal{L}})$$

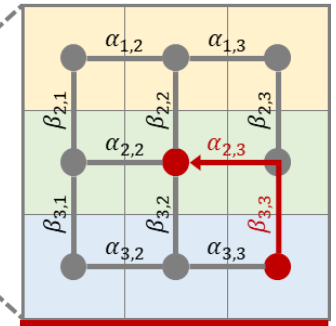


(a) V2H polyline path $\boldsymbol{\mathcal{L}}_{1,1,4,4}$



(b) H2V polyline path $\widetilde{\boldsymbol{\mathcal{L}}}_{1,1,4,4}$





$$L_{5,9} = \alpha_{2,2:3}\beta_{2:3,3}$$

(c) An illustration of the V2H polyline path mask on a 3×3 grid (with a total of 9 tokens).

$$L = \begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}$$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask
  - **Naive Computation**: the polyline mask $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ is large in size and each element $\boldsymbol{\mathcal{L}}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$ require numerous multiplications, resulting in a total complexity of $\mathcal{O}(N^{\frac{5}{2}})$

$$\boldsymbol{\mathcal{L}}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}, \quad where\ i,k = 0,1,\dots,\mathrm{H}-1; \quad k,l = 0,1,\dots,\mathrm{W}-1$$

$$L = \underbrace{\begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}}_{\mathbb{R}^{\mathrm{N}\times\mathrm{N}}}$$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask
  - **Naive Computation**: the polyline mask $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ is large in size and each element $\mathcal{L}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$ require numerous multiplications, resulting in a total complexity of $\mathcal{O}(N^{\frac{5}{2}})$
  - **Efficient Computation**: $\boldsymbol{L}$ can be decomposed as the multiplication of two **sparse matrices**: $\boldsymbol{L} = \boldsymbol{L}^H \times \boldsymbol{L}^V = \hat{\boldsymbol{L}}^H \odot \hat{\boldsymbol{L}}^V$

Theorem 1 (Matrix Decomposition). *For any matrix* $\boldsymbol{M} \in \mathbb{R}^{HW\times HW}$ *and* $\boldsymbol{\mathcal{M}} = \text{fold}(\boldsymbol{M})$, *if for* $\forall i,j,k,l$, $\exists \boldsymbol{A}^i \in \mathbb{R}^{W\times W}$ *and* $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, *s.t.,* $\boldsymbol{\mathcal{M}}_{i,j,k,l} = \left[\boldsymbol{A}^i\right]_{j,l} \times \left[\boldsymbol{B}^l\right]_{i,k}$, *then* $\boldsymbol{M}$ *can be decomposed as:*

$$\boldsymbol{M} = \boldsymbol{M}^A \times \boldsymbol{M}^B = \hat{\boldsymbol{M}}^A \odot \hat{\boldsymbol{M}}^B, \tag{6}$$

where $\boldsymbol{M}^A, \boldsymbol{M}^B, \hat{\boldsymbol{M}}^A, \hat{\boldsymbol{M}}^B \in \mathbb{R}^{HW\times HW}$, *which satisfy*

$$\boldsymbol{M}^A = \text{unfold}(\boldsymbol{\mathcal{M}}^A), \boldsymbol{M}^B = \text{unfold}(\boldsymbol{\mathcal{M}}^B), \text{ s.t., } \boldsymbol{\mathcal{M}}^A_{i,:,k,:} = \begin{cases} \boldsymbol{A}^i & k=i \\ 0 & k\neq i \end{cases}, \boldsymbol{\mathcal{M}}^B_{:,j,:,l} = \begin{cases} \boldsymbol{B}^l & j=l \\ 0 & j\neq l \end{cases}, \tag{7}$$

$$\hat{\boldsymbol{M}}^A = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^A), \ \hat{\boldsymbol{M}}^B = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^B), \ \text{ s.t., } \ \hat{\boldsymbol{\mathcal{M}}}^A_{i,:,k,:} = \boldsymbol{A}^i, \ \hat{\boldsymbol{\mathcal{M}}}^B_{:,j,:,l} = \boldsymbol{B}^l. \tag{8}$$

Corollary 1 (Mask Complexity). *The complexity of directly computing polyline path mask* $\boldsymbol{L}$ *is* $\mathcal{O}(N^{\frac{5}{2}})$, *which can be reduced to* $\mathcal{O}(N^2)$ *by applying Theorem 1, where* $N = H \times W$.

Theorem 1 (Matrix Decomposition). *For any matrix* $\boldsymbol{M} \in \mathbb{R}^{HW\times HW}$ *and* $\boldsymbol{\mathcal{M}} = \text{fold}(\boldsymbol{M})$, *if for* $\forall i,j,k,l$, $\exists \boldsymbol{A}^i \in \mathbb{R}^{W\times W}$ *and* $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, *s.t.,* $\boldsymbol{\mathcal{M}}_{i,j,k,l} = \left[\boldsymbol{A}^i\right]_{j,l} \times \left[\boldsymbol{B}^l\right]_{i,k}$, *then* $\boldsymbol{M}$ *can be decomposed as:*

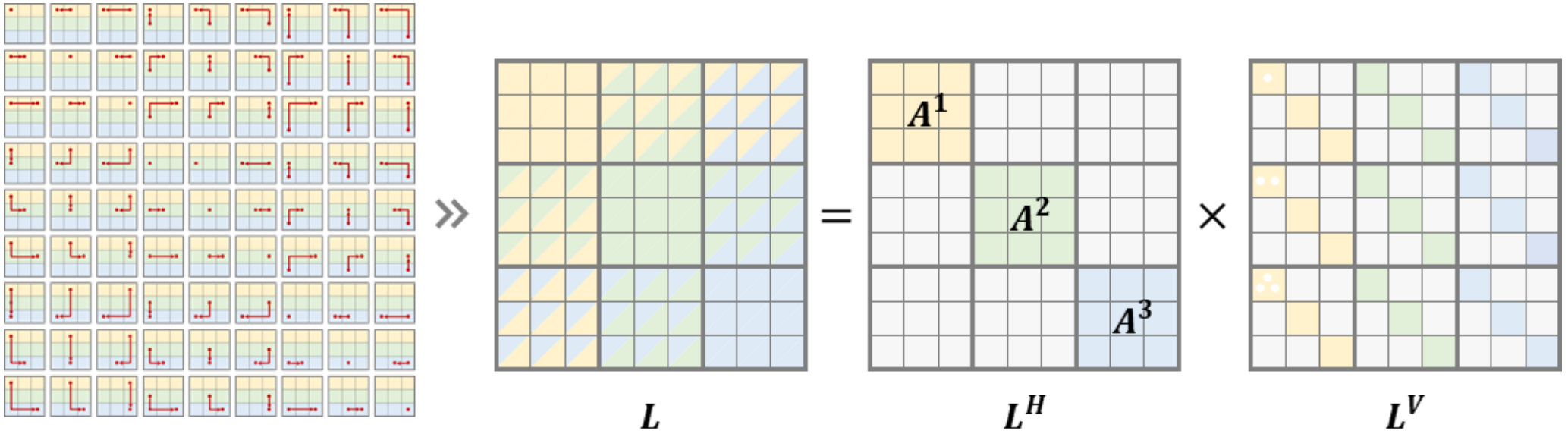
$$\boldsymbol{M} = \boldsymbol{M}^A \times \boldsymbol{M}^B = \hat{\boldsymbol{M}}^A \odot \hat{\boldsymbol{M}}^B, \tag{6}$$

where $\boldsymbol{M}^A, \boldsymbol{M}^B, \hat{\boldsymbol{M}}^A, \hat{\boldsymbol{M}}^B \in \mathbb{R}^{HW\times HW}$, *which satisfy*

$$\boldsymbol{M}^A = \text{unfold}(\boldsymbol{\mathcal{M}}^A), \boldsymbol{M}^B = \text{unfold}(\boldsymbol{\mathcal{M}}^B), \text{ s.t., } \boldsymbol{\mathcal{M}}^A_{i,:,k,:} = \begin{cases} \boldsymbol{A}^i & k=i \\ 0 & k\neq i \end{cases}, \boldsymbol{\mathcal{M}}^B_{:,j,:,l} = \begin{cases} \boldsymbol{B}^l & j=l \\ 0 & j\neq l \end{cases}, \tag{7}$$

$$\hat{\boldsymbol{M}}^A = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^A), \ \hat{\boldsymbol{M}}^B = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^B), \ \text{ s.t., } \ \hat{\boldsymbol{\mathcal{M}}}^A_{i,:,k,:} = \boldsymbol{A}^i, \ \hat{\boldsymbol{\mathcal{M}}}^B_{:,j,:,l} = \boldsymbol{B}^l. \tag{8}$$

The matrix $\boldsymbol{L}$ satisfies the conditions in Theorem 1 with $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$ and $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$.



Polyline Path Scanning

(a) An illustration of the Polyline Path Mask Decomposition

$$
L=\begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}
$$

Complexity: $\mathcal{O}(N^{\frac{5}{2}})$

$$
=\begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
0 & 0 & 0 & & & & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
0 & 0 & 0 & & & & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix}
\times
\begin{bmatrix}
\beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & 0 \\
0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\
0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\
\beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\
0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\
0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\
\beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\
0 & \beta_{3:1,2} & 0 & & & & 0 & \beta_{3:3,2} & 0 \\
0 & 0 & \beta_{3:1,3} & & & & 0 & 0 & \beta_{3:3,3}
\end{bmatrix}
$$

$\boldsymbol{L^H}$: $\mathcal{O}(N^{\frac{3}{2}})$ $\boldsymbol{L^V}$: $\mathcal{O}(N^{\frac{3}{2}})$

$$
=\begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} \\
\alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} \\
\alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} \\
\alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & \alpha_{2,3:1} & \alpha_{2,3:2} & \\
\alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
\alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
\alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix}
\odot
\begin{bmatrix}
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{2:1,1} & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
\beta_{2:1,1} & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
 & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3}
\end{bmatrix}
$$

$\mathcal{O}(N^2)$

Complexity: $\mathcal{O}(N^2)$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask Matrix Multiplication
  - **Naive Computation**: for a rank-N matrix $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ and a vector $\boldsymbol{x} \in \mathbb{R}^{N}$, $\boldsymbol{Lx}$ requires a complexity of $\mathcal{O}(N^2)$
  - **Efficient Computation**: for decomposed polyline path mask, $\boldsymbol{Lx} = \boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{x}$ requires a complexity of $\mathcal{O}(N)$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

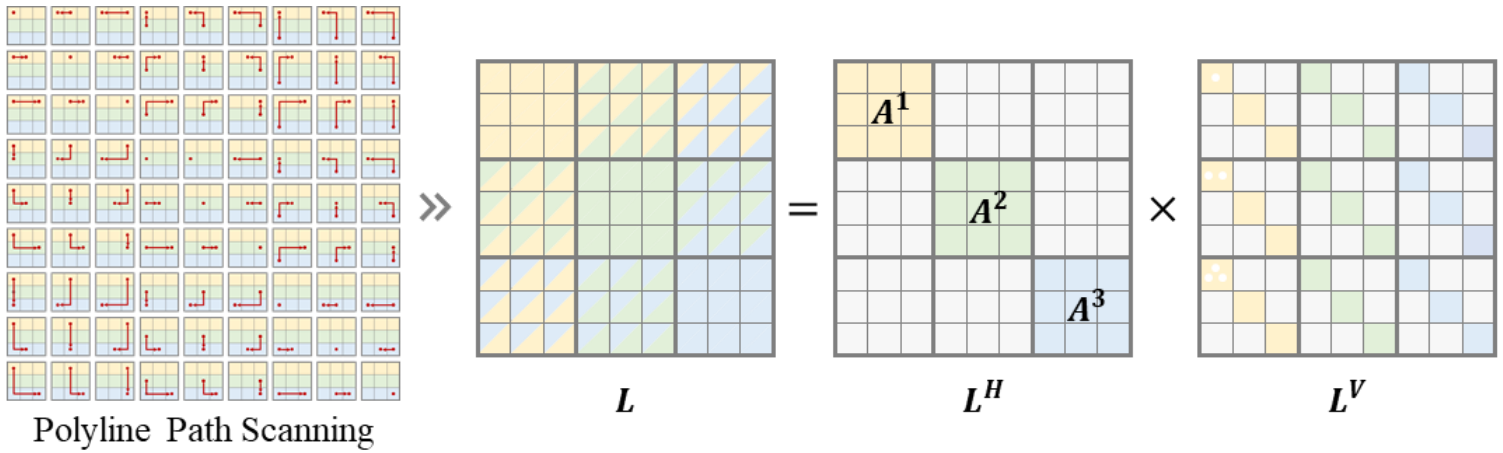
1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

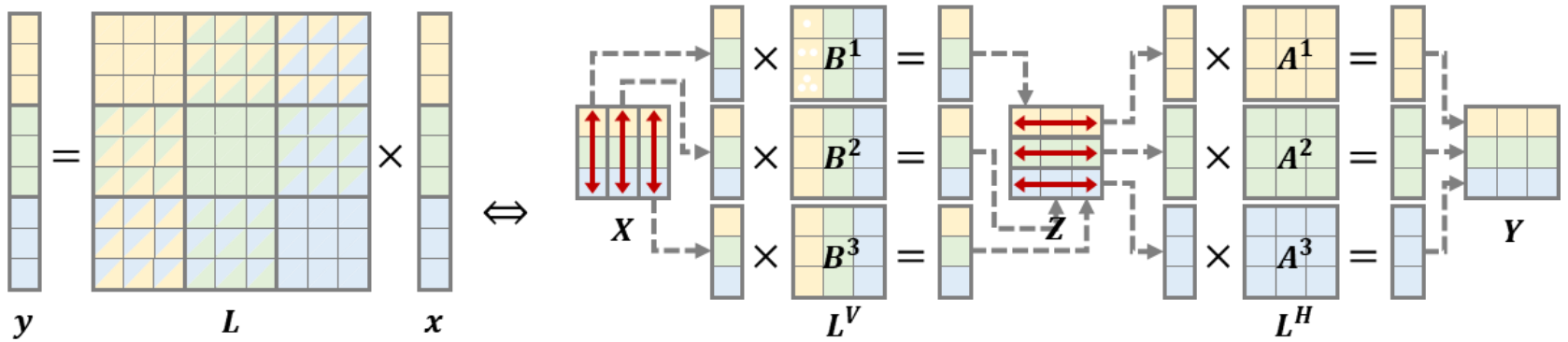
Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.



(a) Polyline Path Mask Decomposition



(b) Polyline Path Mask Multiplication

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \text{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\text{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\text{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1 : W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1 : H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \text{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\text{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\text{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1 : W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1 : H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix} \times \begin{bmatrix}
\beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & 0 \\
0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\
0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\
\beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\
0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\
0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\
\beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\
0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\
0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3}
\end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \text{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\text{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\text{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1 : W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1 : H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix} \boldsymbol{A}^1 & 0 & 0 \\ 0 & \boldsymbol{A}^2 & 0 \\ 0 & 0 & \boldsymbol{A}^3 \end{bmatrix} \times \boldsymbol{B}^1$$

$$\boldsymbol{A}^1 = \begin{bmatrix} \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} \\ \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} \\ \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} \end{bmatrix}$$

$$\boldsymbol{A}^2 = \begin{bmatrix} \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} \\ \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} \\ \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} \end{bmatrix}$$

$$\boldsymbol{A}^3 = \begin{bmatrix} \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\ \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\ \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} \end{bmatrix}$$

$$\boldsymbol{B}^1 = \begin{bmatrix} \beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & 0 \\ 0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\ 0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\ \beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\ 0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\ 0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\ \beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\ 0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\ 0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3} \end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix} \boldsymbol{A}^1 & 0 & 0 \\ \alpha_{2,1:1}\ \alpha_{2,1:2}\ \alpha_{2,1:3} \\ \alpha_{2,2:1}\ \alpha_{2,2:2}\ \alpha_{2,2:3} & \boldsymbol{A}^2 & 0 \\ \alpha_{2,3:1}\ \alpha_{2,3:2}\ \alpha_{2,3:3} \\ 0\ 0\ 0 & \alpha_{3,1:1}\ \alpha_{3,1:2}\ \alpha_{3,1:3} \\ 0\ 0\ 0 & \alpha_{3,2:1}\ \alpha_{3,2:2}\ \alpha_{3,2:3} & \boldsymbol{A}^3 \\ 0\ 0\ 0 & \alpha_{3,3:1}\ \alpha_{3,3:2}\ \alpha_{3,3:3} \end{bmatrix} \times \begin{bmatrix} \beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & \boldsymbol{B}^2 \\ 0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\ 0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\ \beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\ 0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\ 0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\ \beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\ 0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\ 0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3} \end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Y} = \text{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H \times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, and the operator $\text{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\text{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1 : W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1 : H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix} \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ & \boldsymbol{A}^1 & & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & & \boldsymbol{A}^3 & \\ 0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & & & \\ 0 & 0 & 0 & & \boldsymbol{A}^2 & & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\ 0 & 0 & 0 & & & & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} \end{bmatrix} \times \begin{bmatrix} \beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & \boldsymbol{B}^3 \\ 0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\ 0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\ \beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\ 0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\ 0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\ \beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\ 0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\ 0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3} \end{bmatrix}$$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask Matrix Multiplication
  - **Naive Computation**: for a rank-N matrix $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ and a vector $\boldsymbol{x} \in \mathbb{R}^{N}$, $\boldsymbol{Lx}$ requires a complexity of $\mathcal{O}(N^2)$
  - **Efficient Computation**: for decomposed polyline path mask, $\boldsymbol{Lx} = \boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{x}$ requires a complexity of $\mathcal{O}(N)$

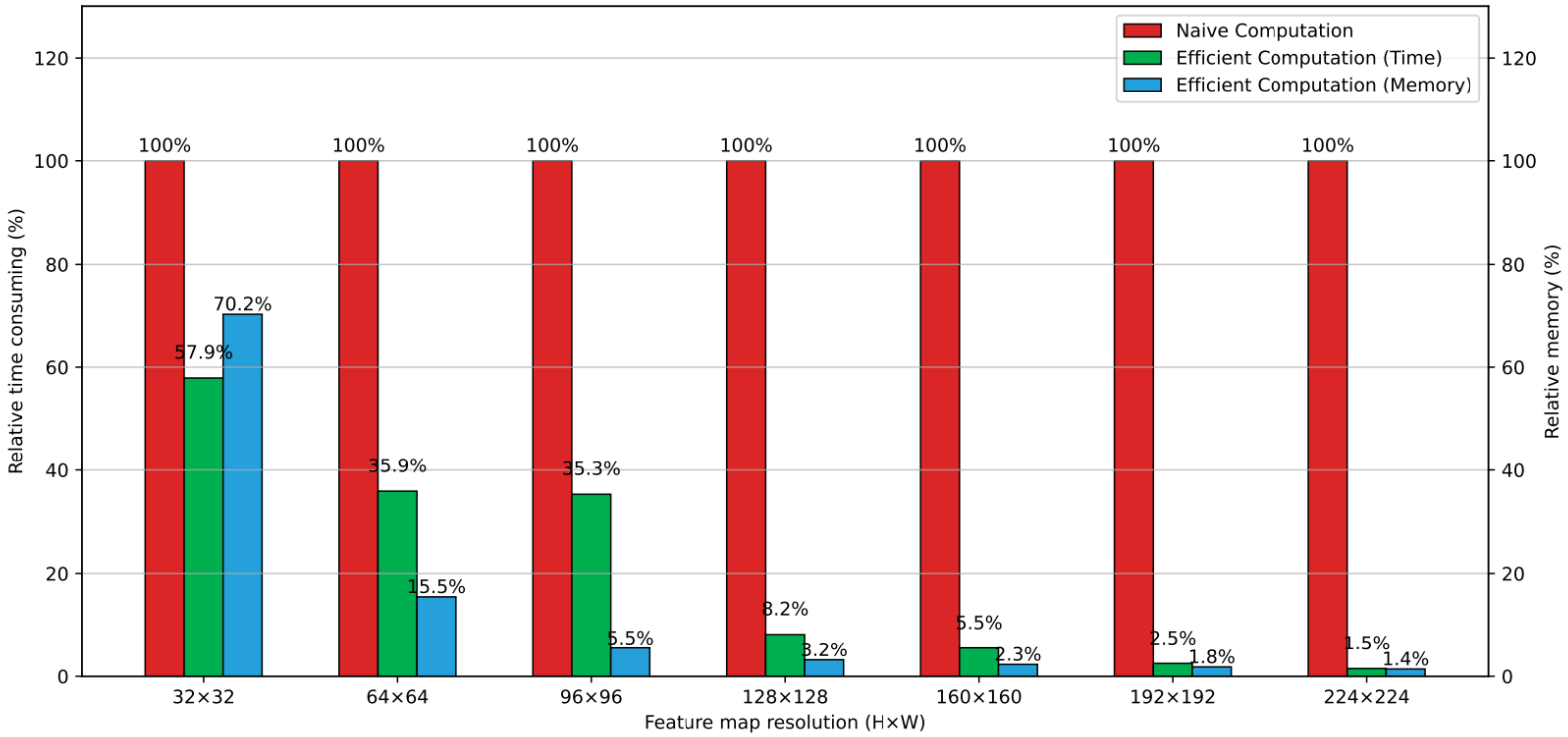


Figure 10: The comparison of the relative time consuming and memory usage between the naive computation and efficient computation (Algorithm 1) of $\boldsymbol{Lx}$.

Remarks. Intuitively, as illustrated in Fig. 4, Algorithm 1 shows that the 2D polyline path scanning on 2D tokens (i.e., $\boldsymbol{L}\boldsymbol{x}$) can be decomposed as the 1D vertical scanning along each column of $\boldsymbol{X}$ (i.e., $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$) followed by the 1D horizontal scanning along each row of $\boldsymbol{Z}$ (i.e., $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$). This equivalence offers an intuitive understanding of the physical meaning of the decomposed polyline path mask $\boldsymbol{L} = \boldsymbol{L}^H \boldsymbol{L}^V$ and enables its natural extension to 3D or higher-dimensional tokens, as detailed in Appendix C.2.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;

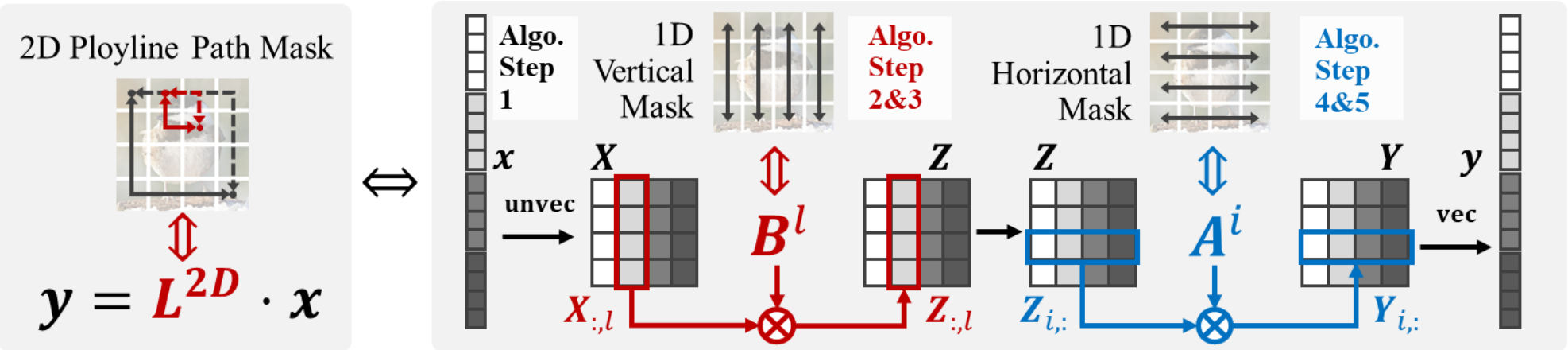
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;

Complexity: $\mathcal{O}(H^2W) \longrightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \longrightarrow \mathcal{O}(HW)$



Remarks. Intuitively, as illustrated in Fig. 4, Algorithm 1 shows that the 2D polyline path scanning on 2D tokens (i.e., $\boldsymbol{L}\boldsymbol{x}$) can be decomposed as the 1D vertical scanning along each column of $\boldsymbol{X}$ (i.e., $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$) followed by the 1D horizontal scanning along each row of $\boldsymbol{Z}$ (i.e., $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$). This equivalence offers an intuitive understanding of the physical meaning of the decomposed polyline path mask $\boldsymbol{L} = \boldsymbol{L}^H \boldsymbol{L}^V$ and enables its natural extension to 3D or higher-dimensional tokens, as detailed in Appendix C.2.

C.2 3D Extension of Polyline Path Mask

Based on the decomposability, we naturally extend the 2D polyline path mask to 3D applications. As illustrated in Fig. 14, the 3D polyline path mask $\boldsymbol{L}^{3D}$ can be decomposed as the multiplication of three 1D structured masks, $\boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{L}^D$, representing the horizontal, vertical, and depth scanning masks, respectively. Specifically, for each token pair $(\boldsymbol{x}_{i,j,k}, \boldsymbol{x}_{l,m,n})$ in the 3D grid, the 3D polyline path mask is defined as:

$$\boldsymbol{\mathcal{L}}^{3D}_{(i,j,k),(l,m,n)} = \alpha_{i,j,k:n} \beta_{i,j:m,n} \gamma_{i:l,m,n}, \tag{40}$$

where $\boldsymbol{\mathcal{L}}^{3D}$ is the tensor form of matrix $\boldsymbol{L}^{3D}$, α, β, and γ are the decay factors along the horizontal, vertical, and depth axes, respectively. Compared to the cross-scanning strategy [30], the 3D polyline path scanning strategy better preserves the adjacency relationships of 3D tokens.

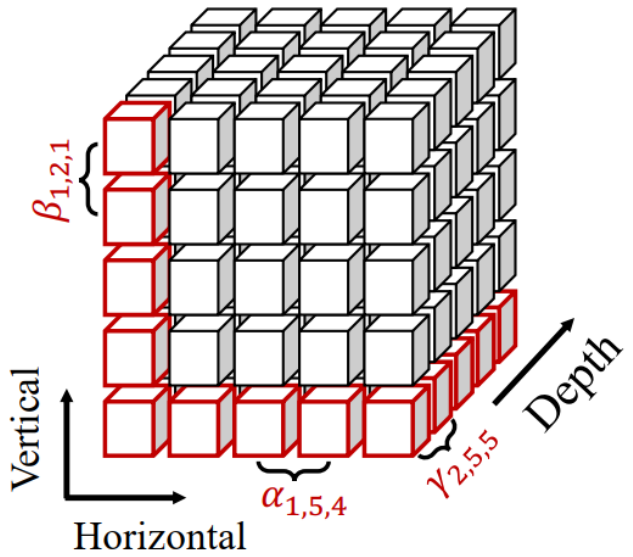


Figure 14: Illustration of the 3D extension of polyline path mask.

5.2 PPMA: Method (Polyline Path Masked Attention)

- Polyline path mask can be integrated into various attention in a plug-and-play manner

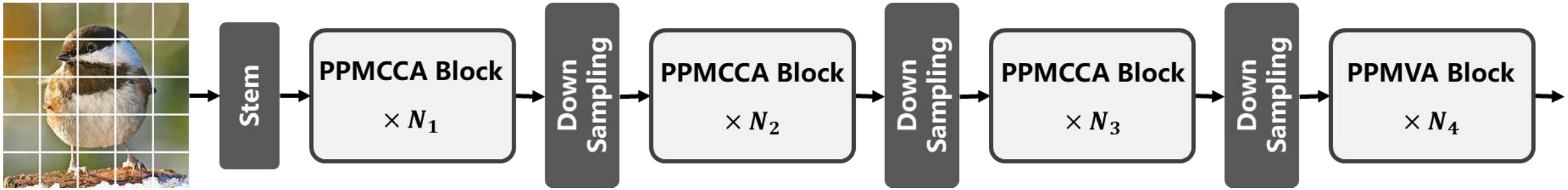
Basic Paradigm: $\mathrm{PPMA}(\boldsymbol{X}) = (\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}) \odot \boldsymbol{L}^{2D})\boldsymbol{V} = (\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}) \odot \boldsymbol{L})\boldsymbol{V} + \left(\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}) \odot \tilde{\boldsymbol{L}}\right)\boldsymbol{V}$

1) Masked **Vanilla self-attention**: $\mathrm{PPMVA}(\boldsymbol{X}) = (\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\top}) \odot \boldsymbol{L}^{2D})\boldsymbol{V}$ *Complexity: $\mathcal{O}(N^2)$*

2) Masked **linear attention**: $\mathrm{PPMLA}(\boldsymbol{X}) = (\boldsymbol{Q}\boldsymbol{K}^{\top} \odot \boldsymbol{L}^{2D})\boldsymbol{V} = \boldsymbol{Q} \star \left(\boldsymbol{L}^{2D} \times (\boldsymbol{K} \star \boldsymbol{V})\right)$ *Complexity: $\mathcal{O}(N)$*

3) Masked **criss-cross attention**: $\mathrm{PPMCCA}(\boldsymbol{X}) = \left(\left(\boldsymbol{S}^{\boldsymbol{H}} \times \boldsymbol{S}^{\boldsymbol{V}}\right) \odot \boldsymbol{L}^{2D}\right)\boldsymbol{V}$ *Complexity: $\mathcal{O}(N^{\frac{3}{2}})$*

4) Masked **decomposed attention** : $\mathrm{PPMDA}(\boldsymbol{X}) = \left((\boldsymbol{S_1} \times \boldsymbol{S_2}) \odot \boldsymbol{L}^{2D}\right)\boldsymbol{V} = \boldsymbol{S_1} \star \left(\boldsymbol{L}^{2D} \times (\boldsymbol{S_2} \star \boldsymbol{V})\right)$



5.2 PPMA: Method (Polyline Path Masked Attention)

- Decomposed Criss-Cross Attention[1]

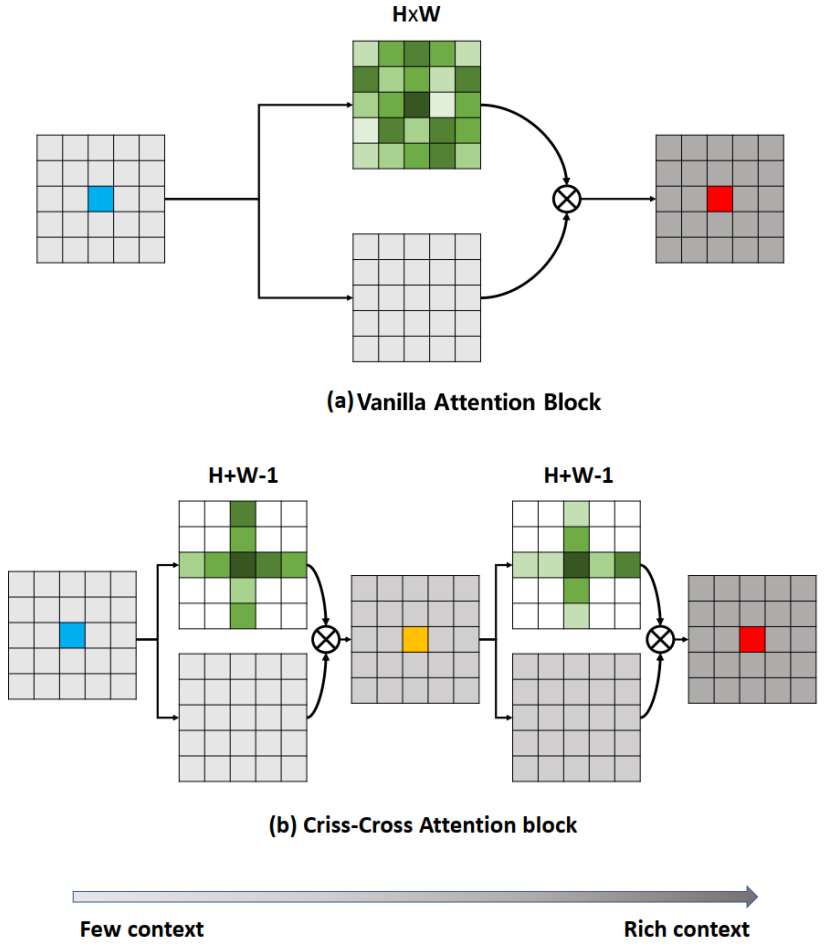
$$\boldsymbol{S^H} = \text{softmax}(\boldsymbol{Q_H K_H^T}) \odot \boldsymbol{L_H}$$

$$\boldsymbol{S^V} = \text{softmax}(\boldsymbol{Q_V K_V^T}) \odot \boldsymbol{L_V}$$

$$\boldsymbol{Y} = 0.5 \times \left(\boldsymbol{S^V}\left(\boldsymbol{S^H V}\right)^\top\right)^\top + 0.5 \times \left(\boldsymbol{S^H}\left(\boldsymbol{S^V V}^\top\right)^\top\right)$$

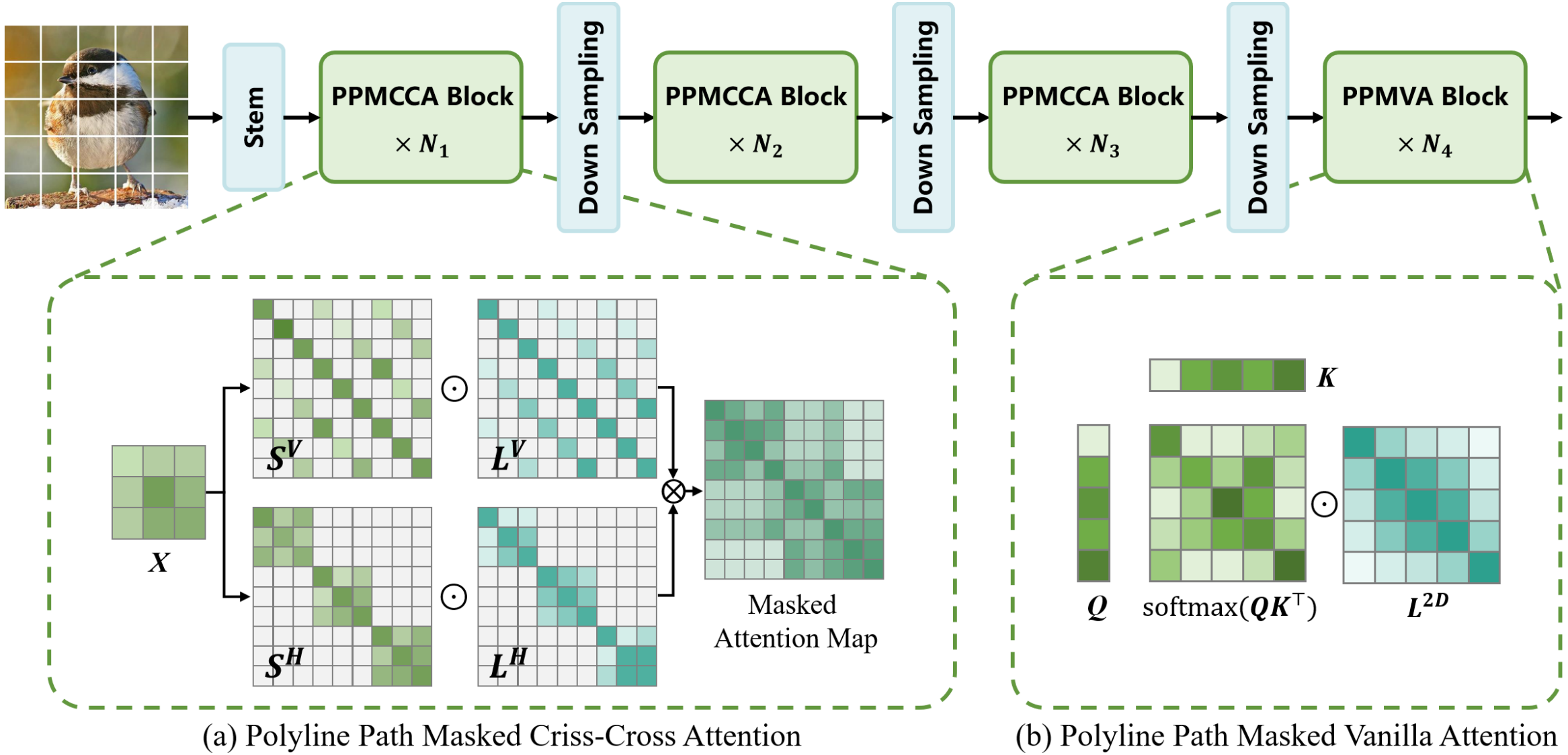
3) Masked **criss-cross attention**: $\text{PPMCCA}(\boldsymbol{X}) = \left(\left(\boldsymbol{S^H} \times \boldsymbol{S^V}\right) \odot \boldsymbol{L}^{2D}\right)\boldsymbol{V}$

$$\begin{aligned}
\left(\left(\boldsymbol{S^H} \times \boldsymbol{S^V}\right) \odot \boldsymbol{L}\right)\boldsymbol{V} &= \left(\left(\boldsymbol{S^H} \times \boldsymbol{S^V}\right) \odot \left(\boldsymbol{L^H} \times \boldsymbol{L^V}\right)\right)\boldsymbol{V} \\
&= \left(\left(\hat{\boldsymbol{S}}^{\boldsymbol{H}} \odot \hat{\boldsymbol{S}}^{\boldsymbol{V}}\right) \odot \left(\hat{\boldsymbol{L}}^{\boldsymbol{H}} \odot \hat{\boldsymbol{L}}^{\boldsymbol{V}}\right)\right)\boldsymbol{V} \\
&= \left(\left(\hat{\boldsymbol{S}}^{\boldsymbol{H}} \odot \hat{\boldsymbol{L}}^{\boldsymbol{H}}\right) \odot \left(\hat{\boldsymbol{S}}^{\boldsymbol{V}} \odot \hat{\boldsymbol{L}}^{\boldsymbol{V}}\right)\right)\boldsymbol{V} \\
&= \left(\left(\boldsymbol{S^H} \odot \boldsymbol{L^H}\right) \times \left(\boldsymbol{S^V} \odot \boldsymbol{L^V}\right)\right)\boldsymbol{V} \\
&= \left(\boldsymbol{S^H} \odot \boldsymbol{L^H}\right) \times \left(\left(\boldsymbol{S^V} \odot \boldsymbol{L^V}\right) \times \boldsymbol{V}\right)
\end{aligned}$$



[1] Huang Z, Wang X, Huang L, et al. Ccnet: Criss-cross attention for semantic segmentation[C], 2019ICCV & 2020TPAMI. (cited:3646)

5.2 PPMA: Method (Polyline Path Masked Attention)



(a) Polyline Path Masked Criss-Cross Attention

(b) Polyline Path Masked Vanilla Attention

5.3 PPMA: Experiments (Image Classification)

Table 1: Image classification performance on the ImageNet-1K validation set.

Model	Arch.	#Param. (M)	FLOPs (G)	Top-1 (%)
RegNetY-1.6G [34]	CNN	11	1.6	78.0
EffNet-B3 [41]	CNN	12	1.8	81.6
Vim-T [57]	SSM	7	1.5	76.1
MSVMamba-M [36]	SSM	12	1.5	79.8
BiFormer-T [56]	Trans.	13	2.2	81.4
NAT-M [16]	Trans.	20	2.7	81.8
SMT-T [29]	Trans.	12	2.4	82.2
RMT-T [10]	Trans.	14	2.5	82.4
PPMA-T		14	2.7	**82.6**
RegNetY-4G [34]	CNN	21	4.0	80.0
ConvNeXt-T [32]	CNN	29	4.5	82.1
EffNet-B4 [41]	CNN	19	4.2	82.9
VMamba-T [30]	SSM	30	4.9	82.6
2DMamba-T [52]	SSM	31	4.9	82.8
GrootVL-T [48]	SSM	30	4.8	83.4
Spatial-Mamba-T [46]	SSM	27	4.5	83.5
MLLA-T [15]	SSM	25	4.2	83.5
Swin-T [31]	Trans.	29	4.5	82.1
CSWin-T [8]	Trans.	23	4.3	82.7
MambaVision-T2 [17]	Trans.	35	5.1	82.7
NAT-T [16]	Trans.	28	4.3	83.2
BiFormer-S [56]	Trans.	26	4.5	83.8
RMT-S [10]	Trans.	27	4.5	84.0
PPMA-S		27	4.9	**84.2**
RegNetY-8G [34]	CNN	39	8.0	81.7
ConvNeXt-S [32]	CNN	50	8.7	83.1
EffNet-B5 [41]	CNN	30	9.9	83.6
VMamba-S [30]	SSM	50	8.7	83.6
2DMamba-S [52]	SSM	50	8.8	83.8
GrootVL-S [48]	SSM	51	8.5	84.2
MLLA-S [15]	SSM	43	7.3	84.4
Spatial-Mamba-S [46]	SSM	43	7.1	84.6
Swin-S [31]	Trans.	50	8.7	83.0
NAT-S [16]	Trans.	51	7.8	83.7
CSWin-B [8]	Trans.	78	15.0	84.2
MambaVision-B [17]	Trans.	98	15.0	84.2
BiFormer-B [56]	Trans.	57	9.8	84.3
iFormer-B [37]	Trans.	48	9.4	84.6
RMT-B [10]	Trans.	54	9.7	84.9
PPMA-B		54	10.6	**85.0**

5.3 PPMA: Experiments (Object Detection)

Table 2: Object detection and instance segmentation performance with Mask R-CNN [18] detector on COCO val2017. FLOPs are calculated with input resolution of 1280×800.

Backbone	#Param. (M)	FLOPs (G)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Vim-T [57]	–	–	45.7	63.9	49.6	39.2	60.9	41.7
MSVMamba-M [36]	32	201	43.8	65.8	47.7	39.9	62.9	42.9
MPViT-XS [25]	30	231	44.2	66.7	48.4	40.4	63.4	43.4
RMT-T [10]	33	218	46.7	68.6	51.6	42.1	65.3	45.2
PPMA-T	33	218	**47.1**	**68.7**	**51.7**	**42.4**	**65.9**	**45.7**
ResNet-50 [19]	44	260	38.2	58.8	41.4	34.7	55.7	37.2
ConvNeXt-T [32]	48	262	44.2	66.6	48.3	40.1	63.3	42.8
MLLA-T [15]	44	255	46.8	69.5	51.5	42.1	66.4	45.0
GrootVL-T [48]	49	265	47.0	69.4	51.5	42.7	66.4	46.0
VMamba-T [30]	50	271	47.3	69.3	52.0	42.7	66.4	45.9
Spatial-Mamba-T [46]	46	216	47.6	69.6	52.3	42.9	66.5	46.2
Swin-T [31]	48	267	43.7	66.6	47.7	39.8	63.3	42.7
CSWin-T [8]	42	279	46.7	68.6	51.3	42.2	65.6	45.4
BiFormer-S [56]	–	–	47.8	69.8	52.3	43.2	66.8	46.5
RMT-S [10]	46	262	48.8	**70.8**	53.4	43.6	**67.4**	**47.3**
PPMA-S	46	263	**49.2**	70.7	**54.0**	**43.8**	**67.4**	47.1
ResNet-101 [19]	63	336	40.4	61.1	44.2	36.4	57.7	38.8
ConvNeXt-S [32]	70	348	45.4	67.9	50.0	41.8	65.2	45.1
GrootVL-S [48]	70	341	48.6	70.3	53.5	43.6	67.5	47.1
VMamba-S [30]	70	349	48.7	70.0	53.4	43.7	67.3	47.0
Spatial-Mamba-S [46]	63	315	49.2	70.8	54.2	44.0	67.9	47.5
MLLA-S [15]	63	319	49.2	71.5	53.9	44.2	68.5	47.2
Swin-S [31]	69	359	45.7	67.9	50.4	41.1	64.9	44.2
CSWin-S [8]	54	342	47.9	70.1	52.6	43.2	67.1	46.2
BiFormer-B [56]	–	–	48.6	70.5	53.8	43.7	67.6	47.1
RMT-B [10]	73	373	50.7	72.0	55.7	45.1	69.2	49.0
PPMA-B	73	374	**51.1**	**72.5**	**55.9**	**45.5**	**69.7**	**49.1**

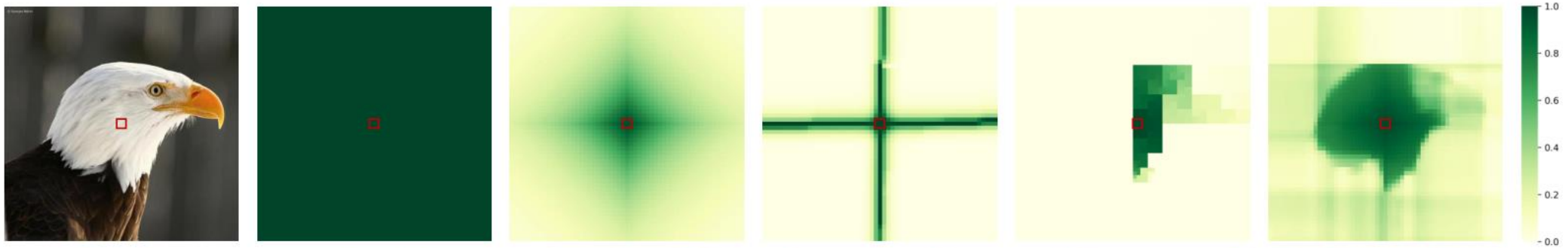
5.3 PPMA: Experiments (Semantic Segmentation)

Table 3: Semantic segmentation performance with UPerNet [47] segmentor on ADE20K val set. 'SS' and 'MS' represent single-scale and multi-scale testing, respectively.

Backbone	#Param. (M)	FLOPs (G)	mIoU(%) SS	mIoU(%) MS
LocalVim-T [21]	36	181	43.4	44.4
MSVMamba-M [36]	42	875	45.1	45.4
NAT-M [16]	50	900	45.1	46.4
RMT-T [10]	43	977	48.0	48.8
PPMA-T	43	983	**48.7**	**49.1**
ResNet-50 [19]	67	953	42.1	42.8
ConvNeXt-T [32]	60	939	46.0	46.7
VMamba-T [30]	62	949	48.0	48.8
2DMamba-T [52]	62	950	48.6	49.3
GrootVL-T [48]	60	941	48.5	49.4
Spatial-Mamba-S [46]	57	936	48.6	49.4
Swin-T [31]	60	945	44.4	45.8
MambaVision-T [17]	55	945	46.6	–
NAT-T [16]	58	934	47.1	48.4
CSWin-S [8]	60	959	49.3	50.7
BiFormer-S [56]	–	–	49.8	50.8
RMT-S [10]	56	937	49.8	49.7
PPMA-S	56	984	**51.1**	**52.0**
ResNet-101 [19]	85	1030	42.9	44.0
ConvNeXt-S [32]	82	1027	48.7	49.6
VMamba-S [30]	82	1028	50.6	51.2
Spatial-Mamba-S [46]	73	992	50.6	51.4
GrootVL-S [48]	82	1019	50.7	51.7
Swin-S [31]	81	1039	47.6	49.5
NAT-S [16]	82	1010	48.0	49.5
MambaVision-S [17]	84	1135	48.2	–
CSWin-S [8]	65	1027	50.4	51.5
BiFormer-B [56]	–	–	51.0	51.7
RMT-B [10]	83	1051	52.0	52.1
PPMA-B	83	1137	**52.3**	**53.0**

5.3 PPMA: Experiments (Ablation Study)

- Ablation study on the polyline path mask design



(a) Input Image (b) W/o Mask (c) RMT Decay Mask (d) Cross Scan Mask (e) Hilbert Scan Mask (f) Polyline Path Mask

Figure 6: Illustration of various structured masks.

Table 9: Ablation study of structured mask designs in PPMA-T on ImageNet-1K and ADE20K.

Structured Mask	#Param. (M)	FLOPs (G)	Top-1 (%)	mIoU SS (%)
Baseline (w/o mask)	14.33	2.65	82.28	47.78
+ RMT Decay Mask	14.33	2.65	82.35	48.01
+ Cross Scan Mask	14.34	2.71	82.44	48.14
+ V2H Polyline Path Mask	14.34	2.71	82.44	48.57
+ 2D Polyline Path Mask	14.34	2.71	**82.60**	**48.73**
Shared Decay factors ($\alpha_{i,j} = \beta_{i,j}$)	14.33	2.71	82.37	48.27
Different Decay factors ($\alpha_{i,j} \neq \beta_{i,j}$)	14.34	2.71	**82.60**	**48.73**

5.3 PPMA: Experiments (Visualization)

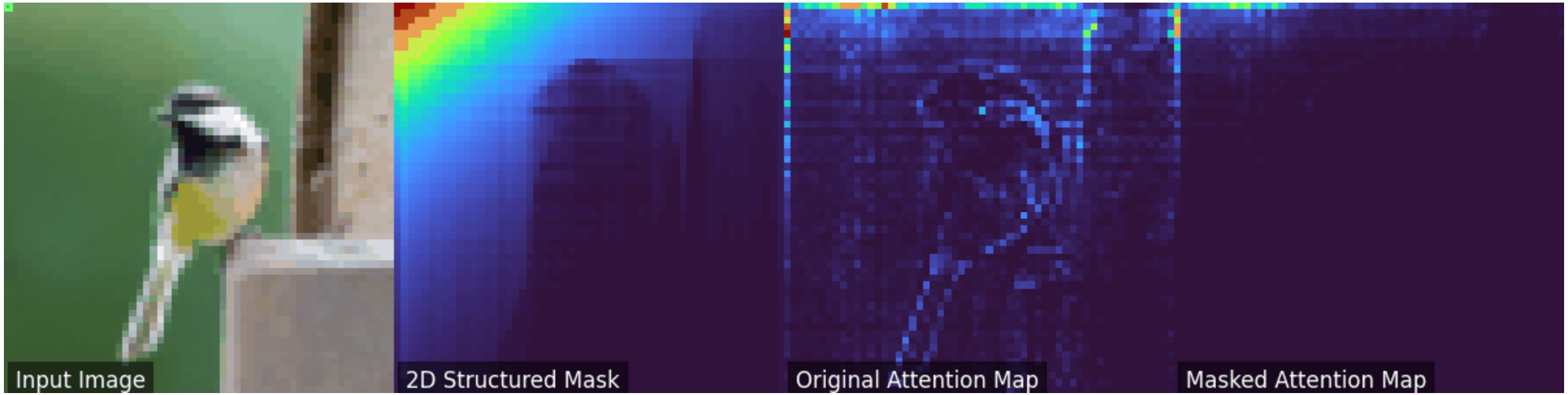
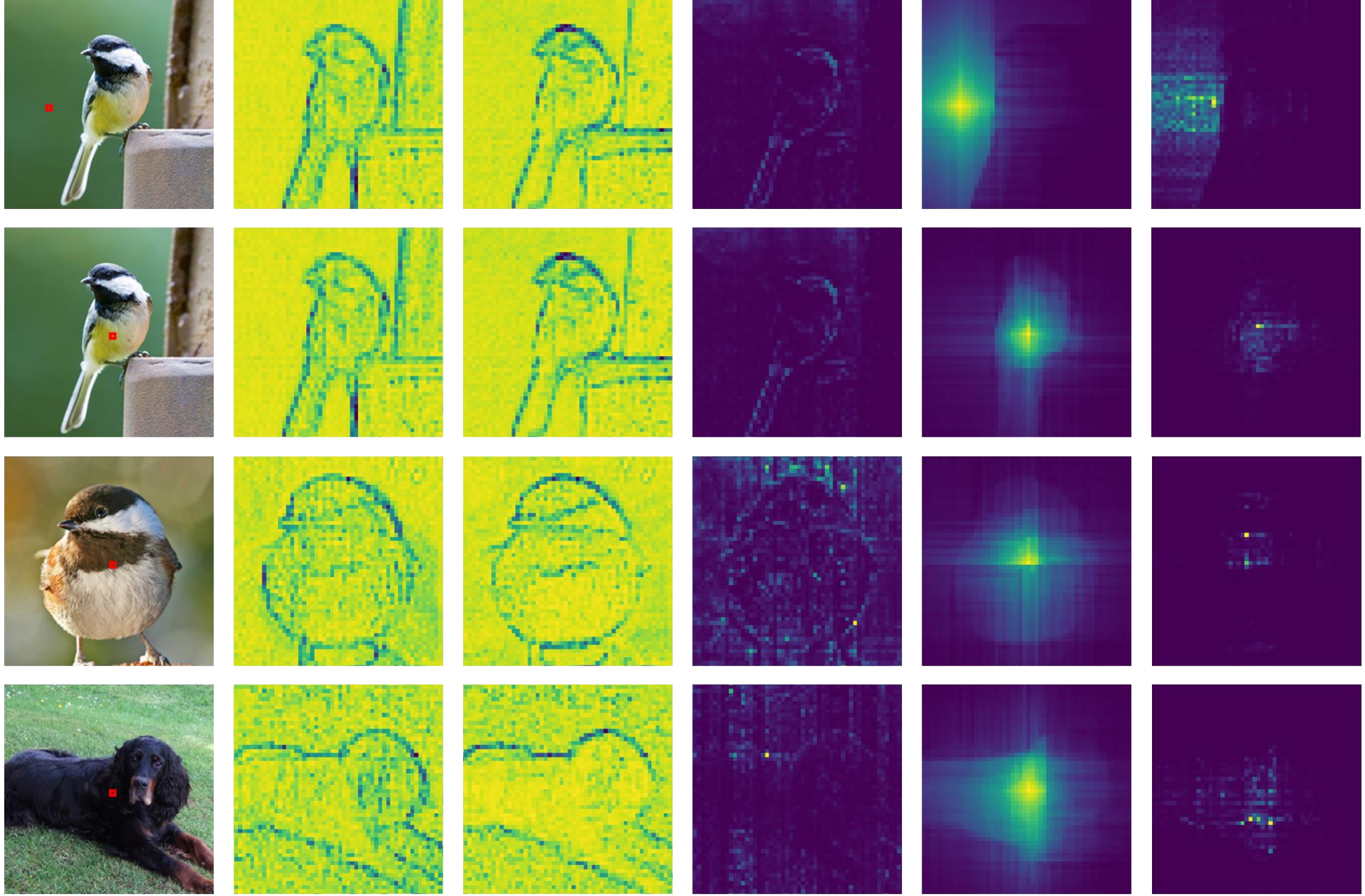
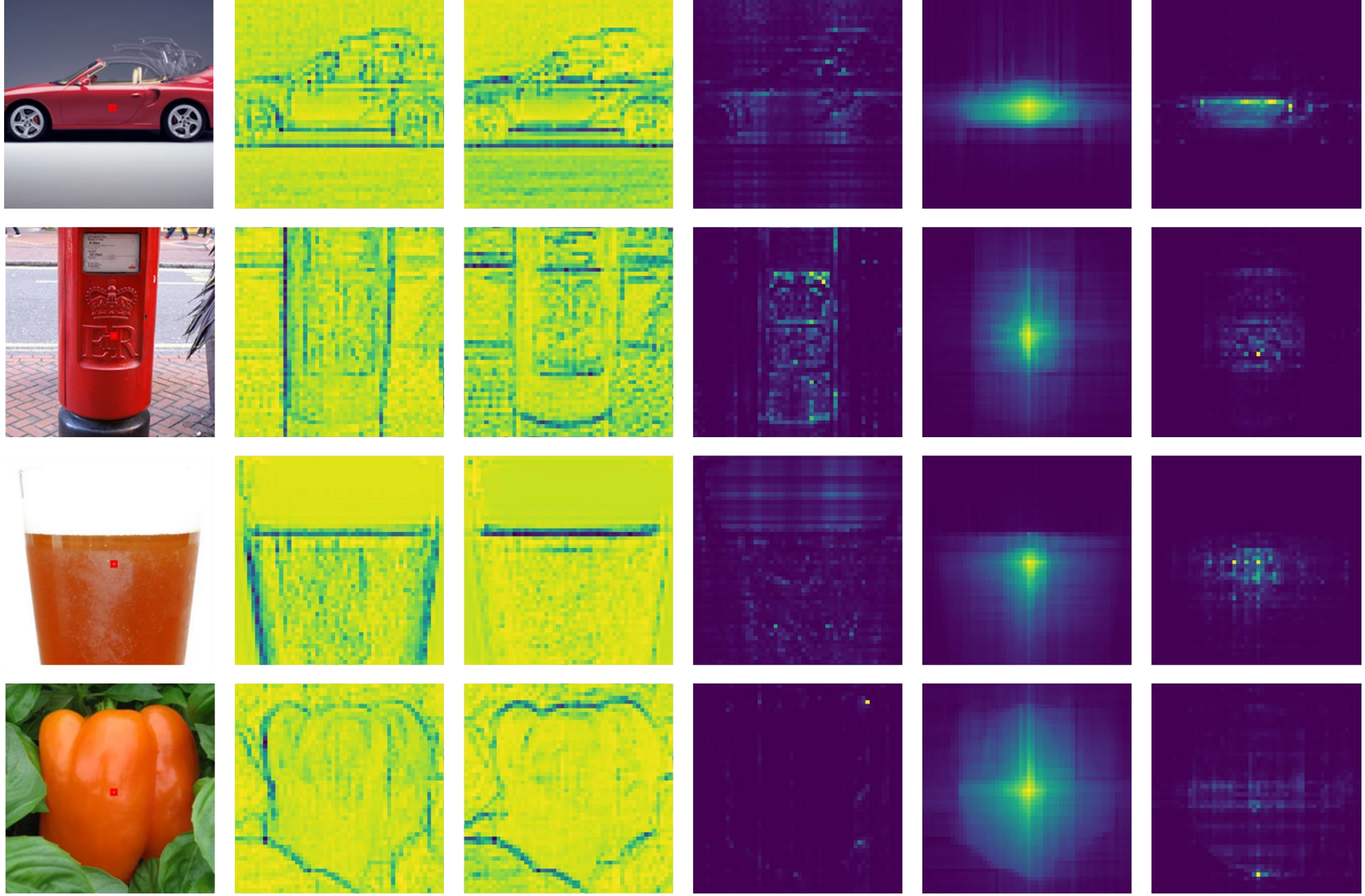


Figure 8: Visualizations of the Polyline Path Masked Attention



(a) Input Image | (b) Horizontal Decay Factor $\boldsymbol{\alpha}$ | (c) Vertical Decay Factor $\boldsymbol{\beta}$ | (d) Original Attention Map softmax($\boldsymbol{QK}^{\top}$) | (e) Polyline Path Mask $\boldsymbol{L}^{2D}$ | (f) Polyline Path Masked Attention Map



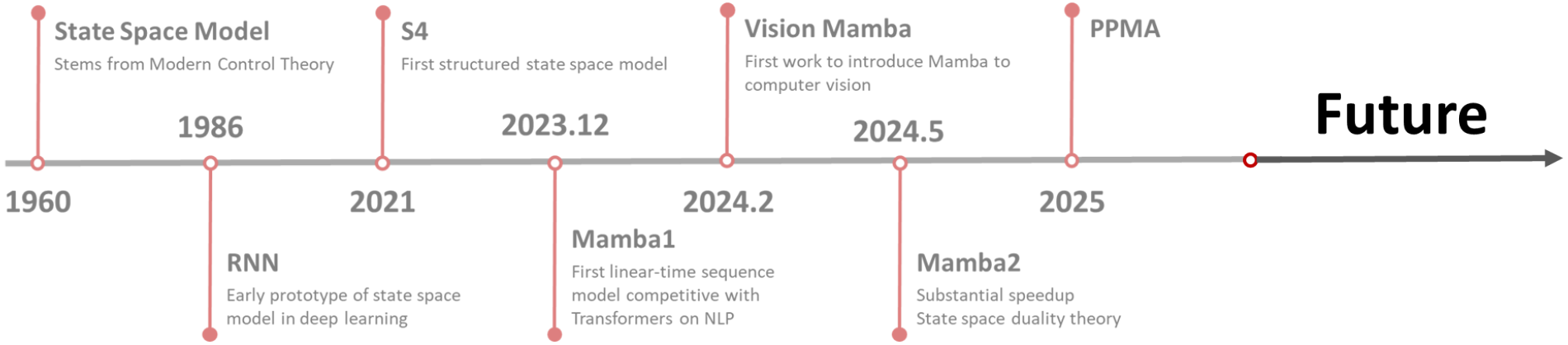
(a) Input Image | (b) Horizontal Decay Factor $\boldsymbol{\alpha}$ | (c) Vertical Decay Factor $\boldsymbol{\beta}$ | (d) Original Attention Map softmax($\boldsymbol{QK}^{\top}$) | (e) Polyline Path Mask $\boldsymbol{L}^{2D}$ | (f) Polyline Path Masked Attention Map

What is next for Mamba?

- Stable and scalable linear-time foundational model remains a worthwhile subject
- Hybrid architecture maybe the future

9.2.3 Hybrid Models: Combining SSD Layer with MLP and Attention

Recent and concurrent work (Dao, D. Y. Fu, et al. 2023; De et al. 2024; Glorioso et al. 2024; Lieber et al. 2024) suggests that a hybrid architecture with both SSM layers and attention layers could improve the model quality over that of a Transformer, or a pure SSM (e.g., Mamba) model, especially for in-context learning. We explore the different ways that SSD layers can be combined with attention and MLP to understand the benefits of each. Empirically we find that having around 10% of the total number of layers being attention performs best. Combining SSD layers, attention layers, and MLP also works better than either pure Transformer++ or Mamba-2.



西安交通大学
XI'AN JIAOTONG UNIVERSITY

Thanks!