Mamba: Linear-Time Sequence Modeling with Selective State Spaces

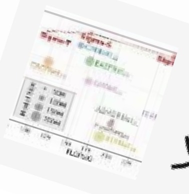
Arxiv 2312

Albert Gu, Tri Dao

Minghao Liu
2024.4.14

MambaMixer：一种高效选择性状态空间模型，它使用跨 token 和通道的双重选择机制，称为选择性 token 和通道混合器，在视觉和时序预测任务上表现出色！代码即将开源👍 单位：康奈尔大学等

Sigma：第一个成功将Mamba应用于多模态语义分割的新工作，采用Siamese 编码器并创新 Mamba 融合机制，有效地从不同模态中选择重要信息，性能表现SOTA！代码已开源👍



Mamba再下一城！Sigma：多模态语义分割的孪生Mamba网络

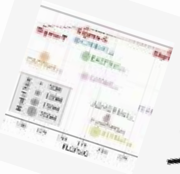
态空间模型，它使用跨token和通道的
通道混合器，在视觉和时序预测任务
：康奈尔大学等

双重
上表现

让Mamba涨点！本文在Mamba层和自注意力层之间建立了重要的联系，直接将高效的Mamba层与Transformer层连接起来，还提出Mamba第一个可解释性技术！代码刚开源👍

涨点！Mamba 模型的隐藏注意力

Sigma：第一个
Siamese 编码器
信息，性能表现SC



Mamba再
义分割的孪生Mamba网络

级！Sigma：多模态语

n和通道的
序预测任务

从不同模态中选择重要
，采用
通道混合器
：康奈尔大学等

双重
上表现

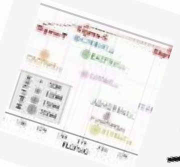
InsectMamba：第一个将Mamba应用于害虫分类的网络，集成了SSM，CNN，多头自注意力机制和MLP，在五个害虫分类数据集上性能表现出色👍

InsectMamba：基于状态空间模型的害虫分类

本文在Mamba层和自注意力层之间建立了重要的联
直接将高效的Mamba层与Transformer层连接起来，还提出Mamba
第一个可解释性技术！代码刚开源👍

涨点！Mamba 模型的隐藏注意力

码器
性能表现



Mamba再
级！Sigma：多模态语
义分割的孪生Mamba网络

和通道的
序预测任务

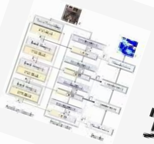
采用
通道混合
不同模态中选择重要
康奈尔大学等

双
上表现

InsectMamba：第一个将Mamba应用于害虫分类的网络，集成了SSM、CNN，多头自注意力机制和MLP，在五个害虫分类数据集上表现出色👍

InsectMamba：基于状态空间模型的害虫分类

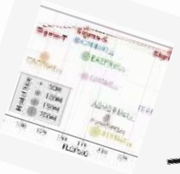
据称，RS3Mamba：第一个专门为遥感图像语义分割设计的视觉Mamba，利用VSS块，并引入协作完成模块（CCM），性能表现出色！代码即将开源👍
单位：香港中文大学，武汉科技大学



RS3Mamba：遥感图像语义分割的视觉状态空间模型

本文在Mamba层和自注意
直接将高效的Mamba层与Transformer层连接
第一个可解释性技术！代码刚开源👍

涨点！Mamba 模型的隐藏注意力



Mamba再

Sigma：多模态语义分割的孪生Mamba网络

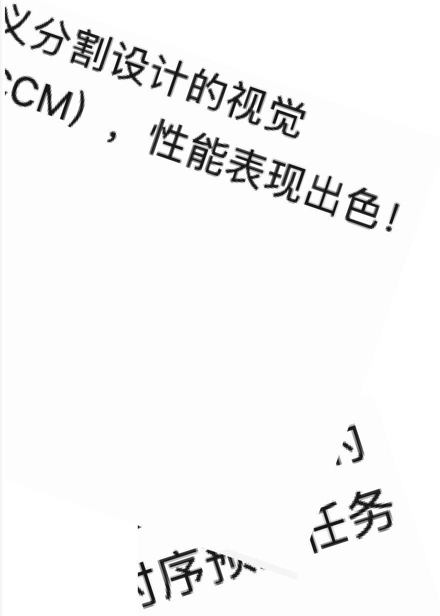
从不同模态中选择重要
采用
通道混合器
康奈尔大学等
性能表现SO
任务
时序预
双重
上表现

Albert Gu @_albertgu · 2023年12月5日

Quadratic attention has been indispensable for information-dense modalities such as language... until now.

Announcing Mamba: a new SSM arch. that has linear-time scaling, ultra long context, and most importantly--outperforms Transformers everywhere we've tried.

With @tri_dao 1/



Background

- Foundation models
- Large models pretrained on massive data then adapted for downstream tasks
- Backbone : sequence models

Transformer

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

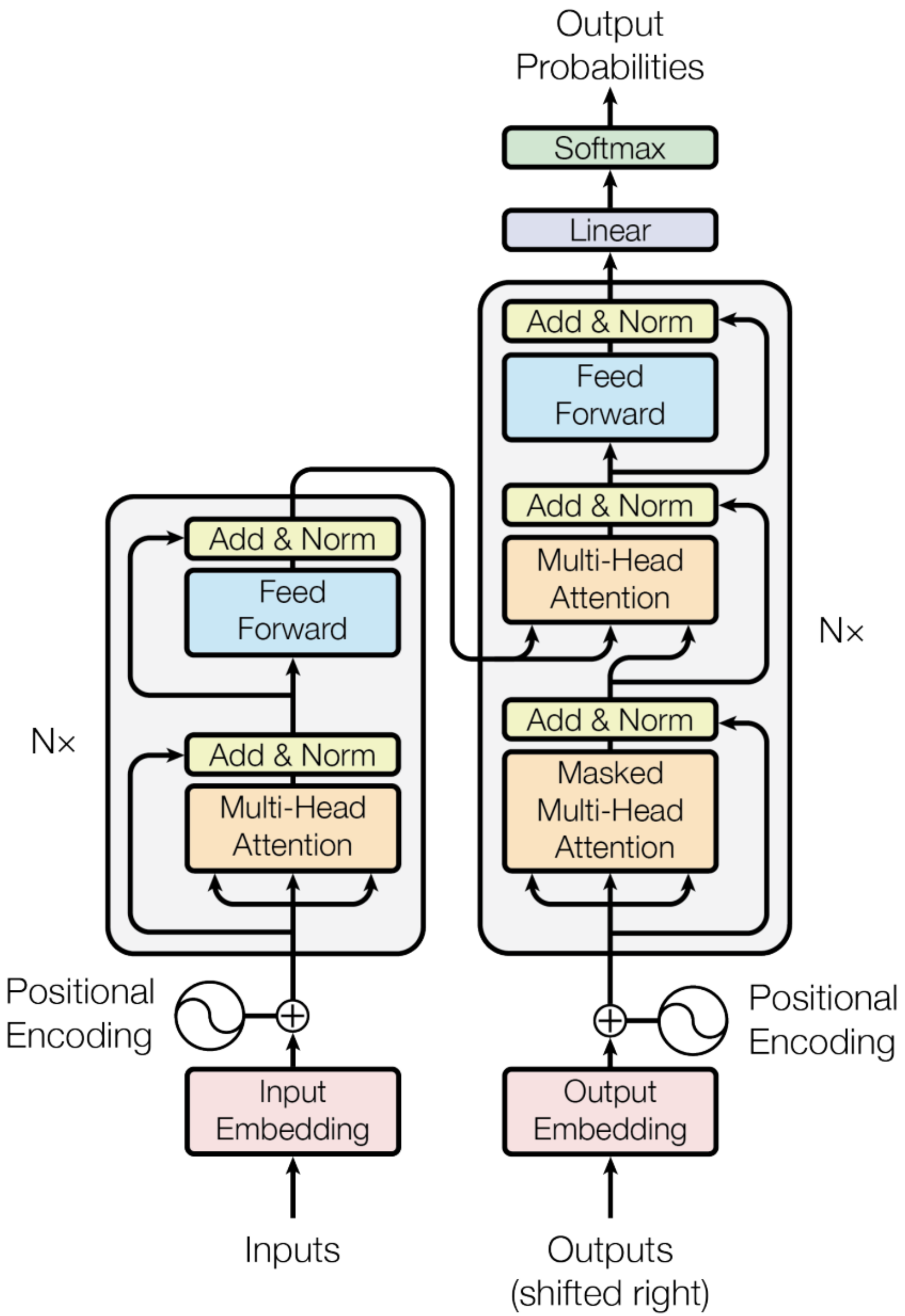
Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[* †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[* ‡]
illia.polosukhin@gmail.com

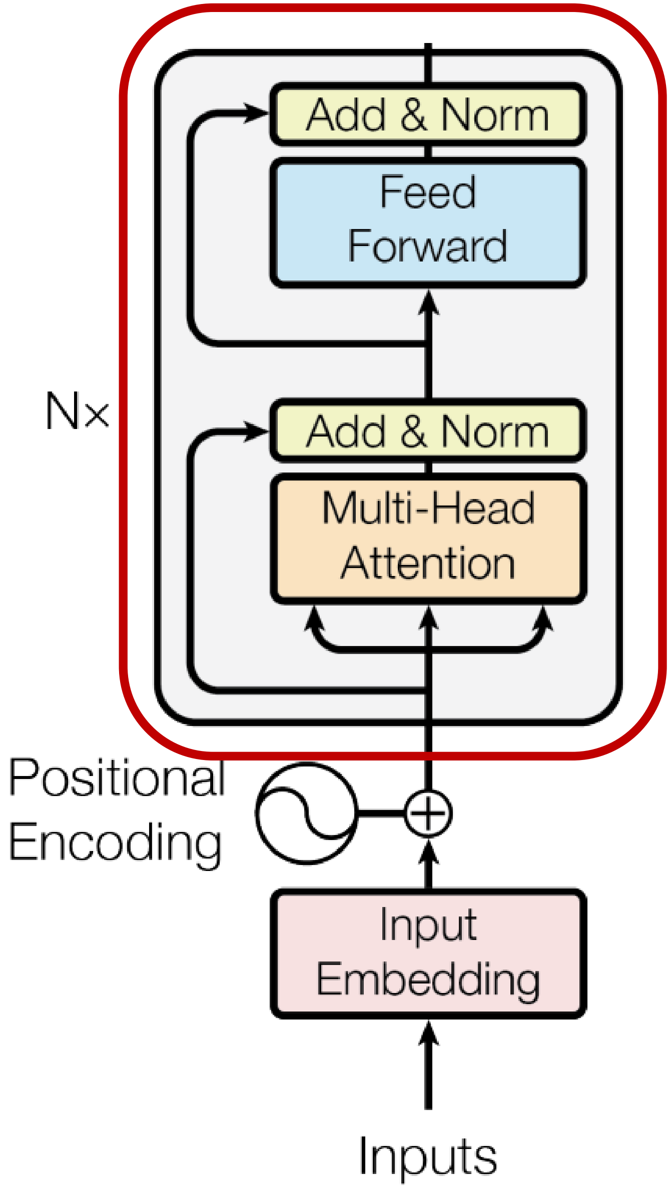


Transformer

- Encoder:
- Multi-Head Self-Attention

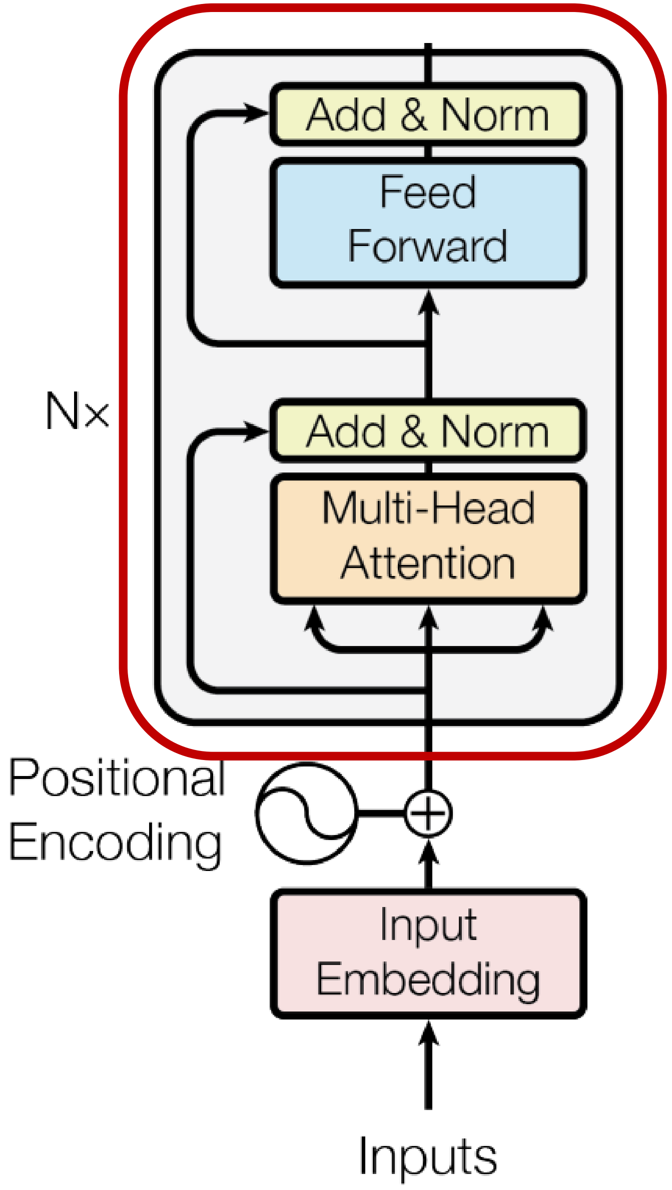
- Feed Forward

- Residual Connection & Layer Norm



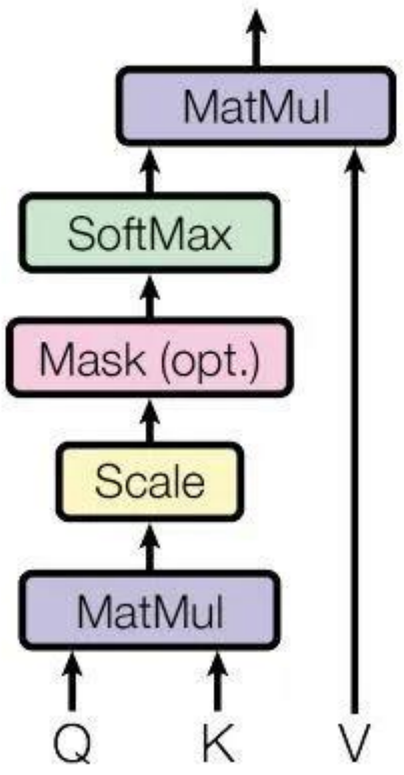
Transformer

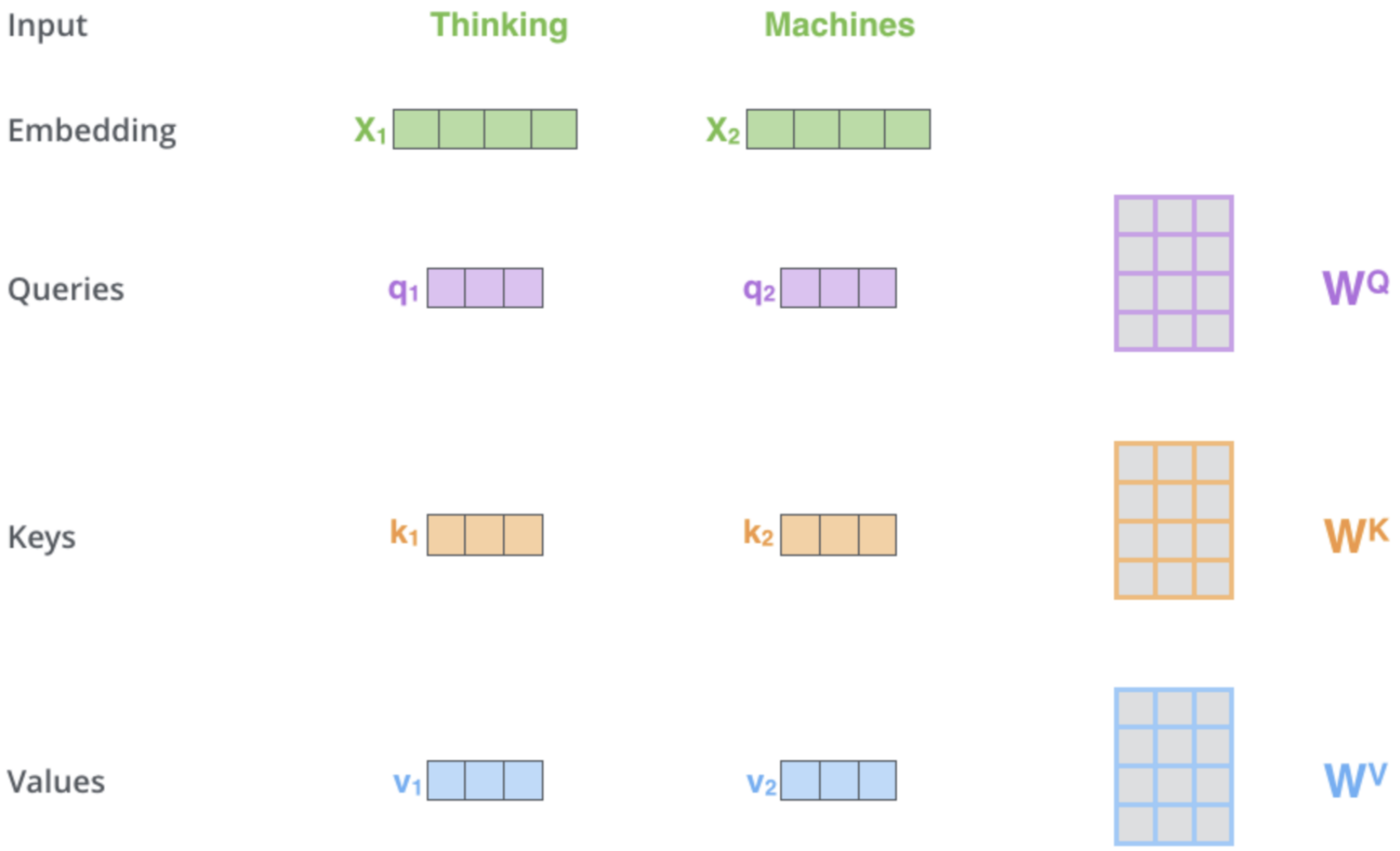
- Encoder:
- Multi-Head Self-Attention
- Feed Forward
- Residual Connection & Layer Norm



Self-Attention

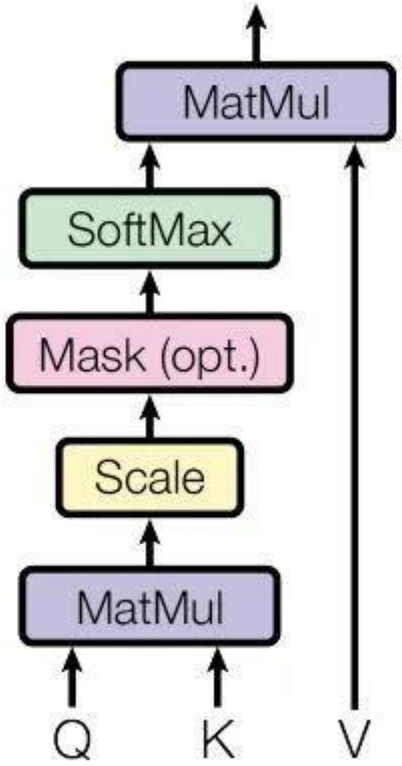
Scaled Dot-Product Attention





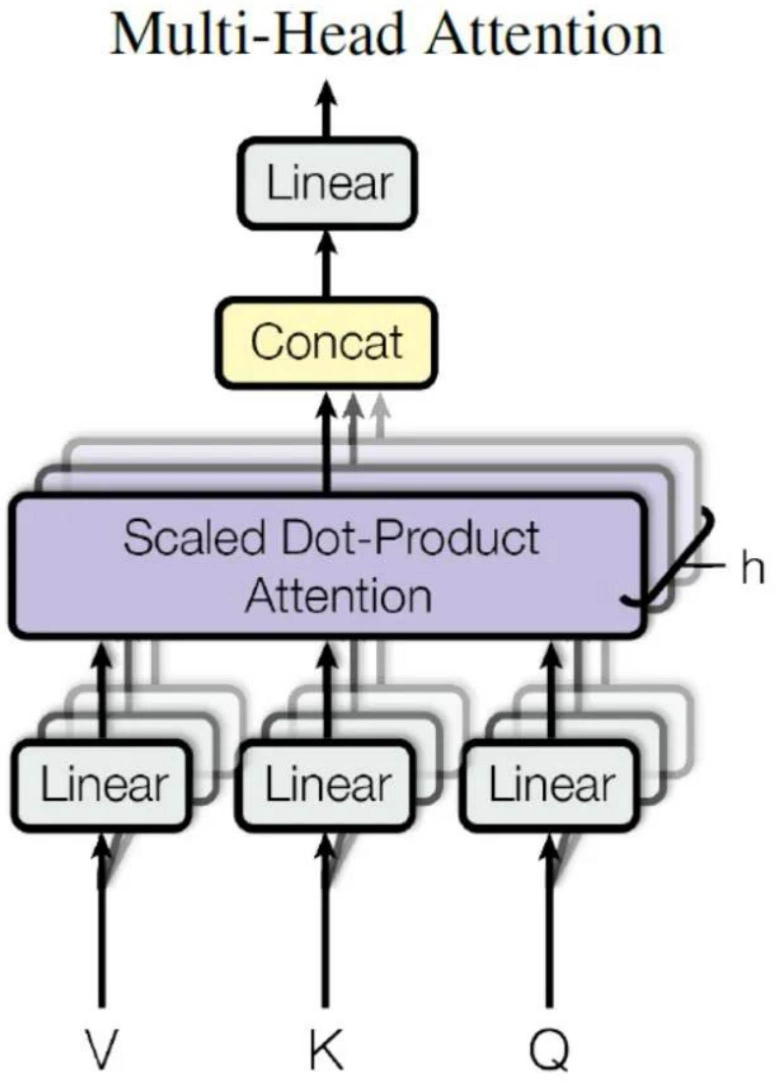
Self-Attention

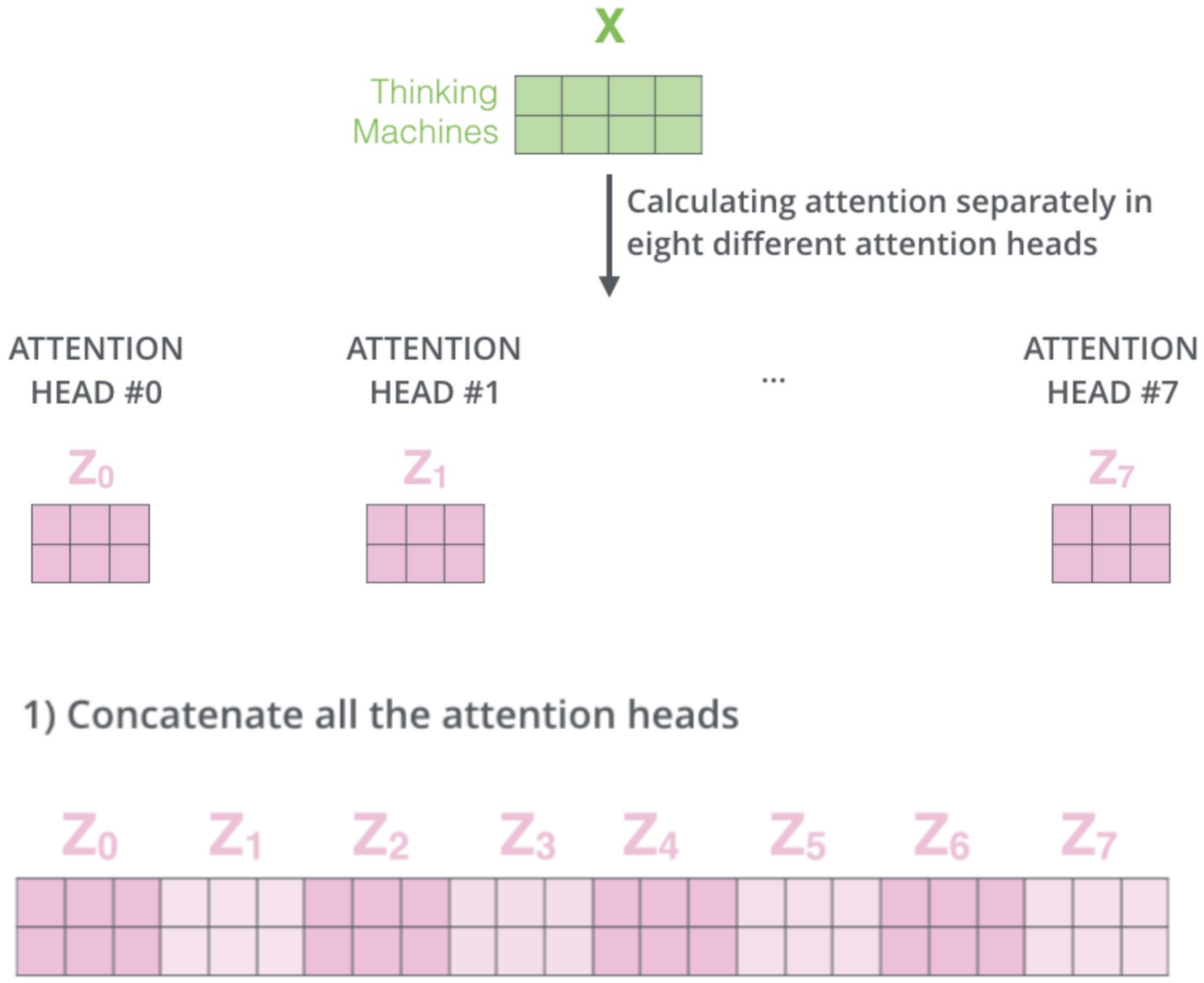
Scaled Dot-Product Attention



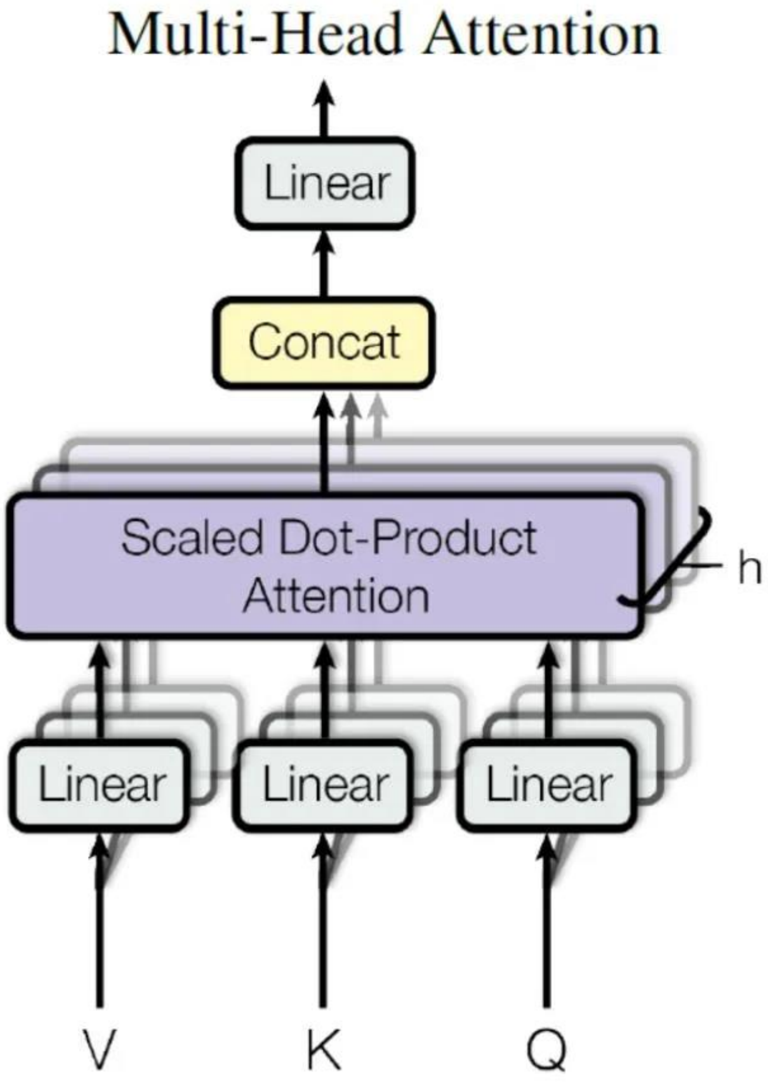
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Self-Attention





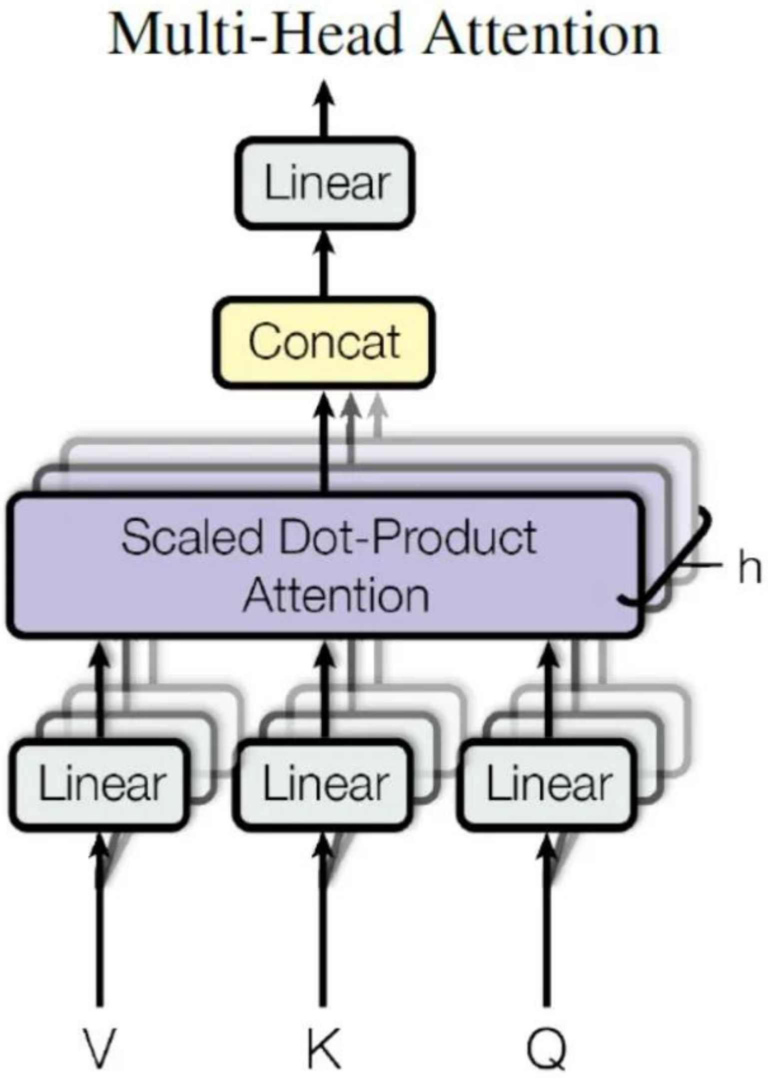
Multi-Head Self-Attention



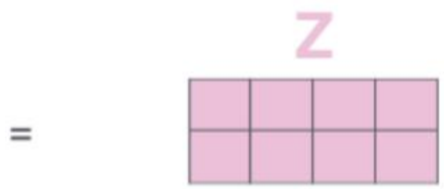
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

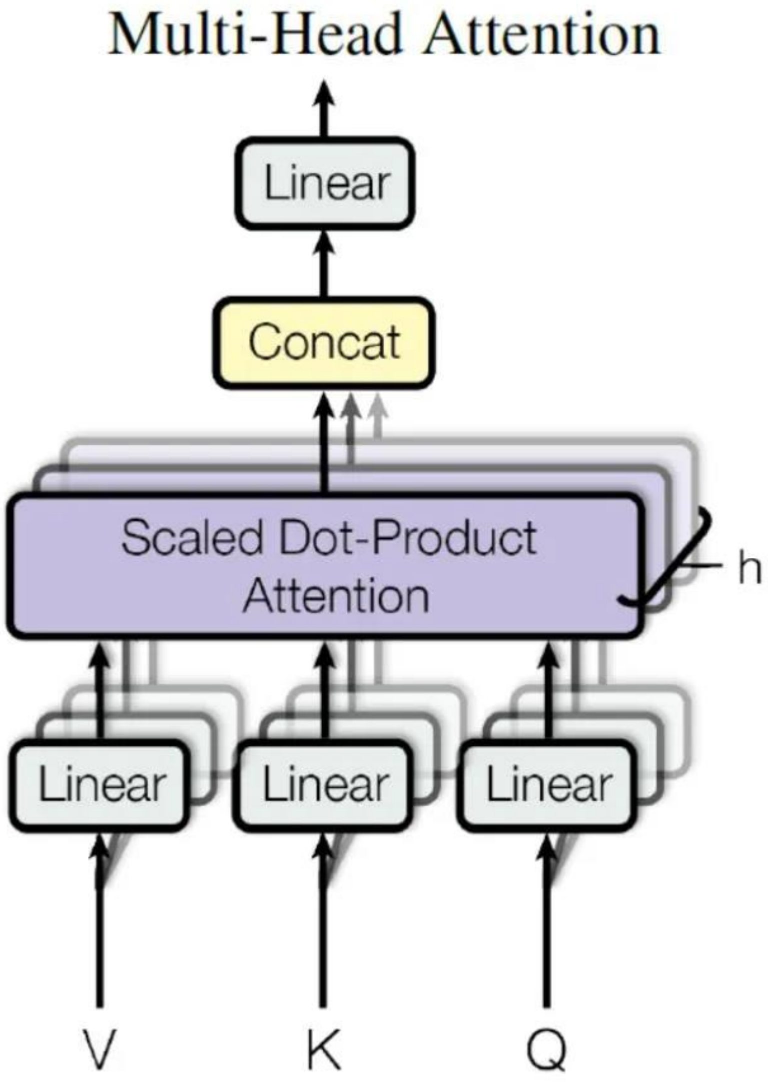
Multi-Head Self-Attention



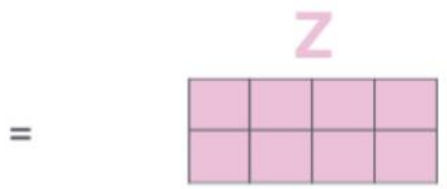
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Multi-Head Self-Attention



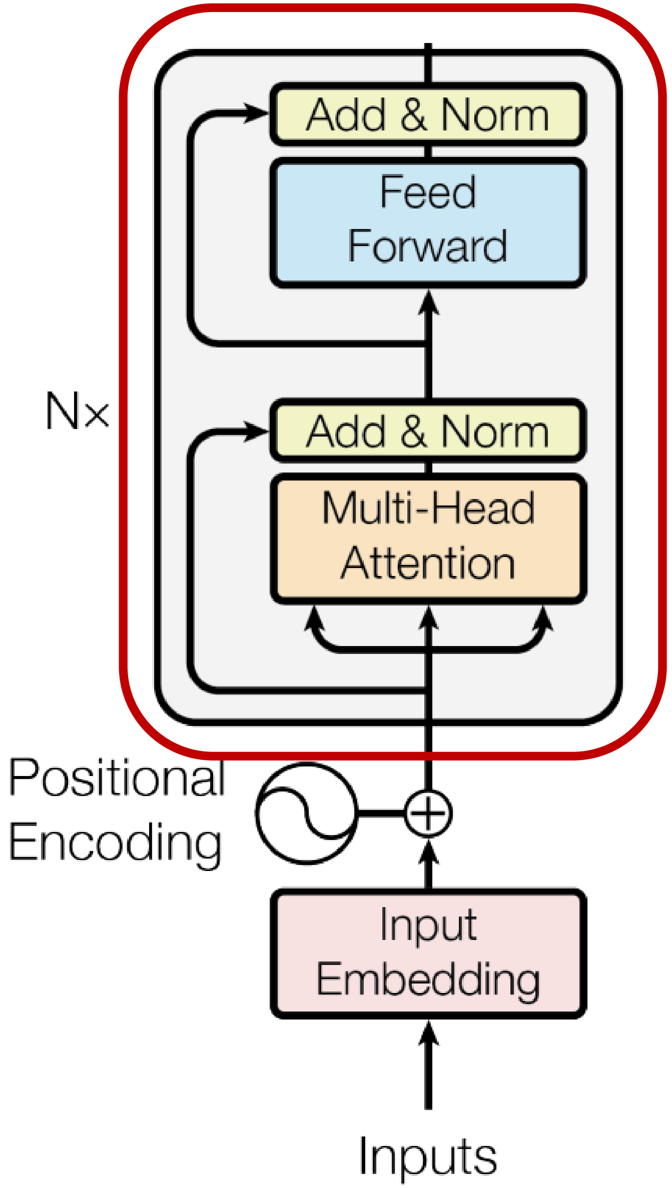
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



‘Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.’

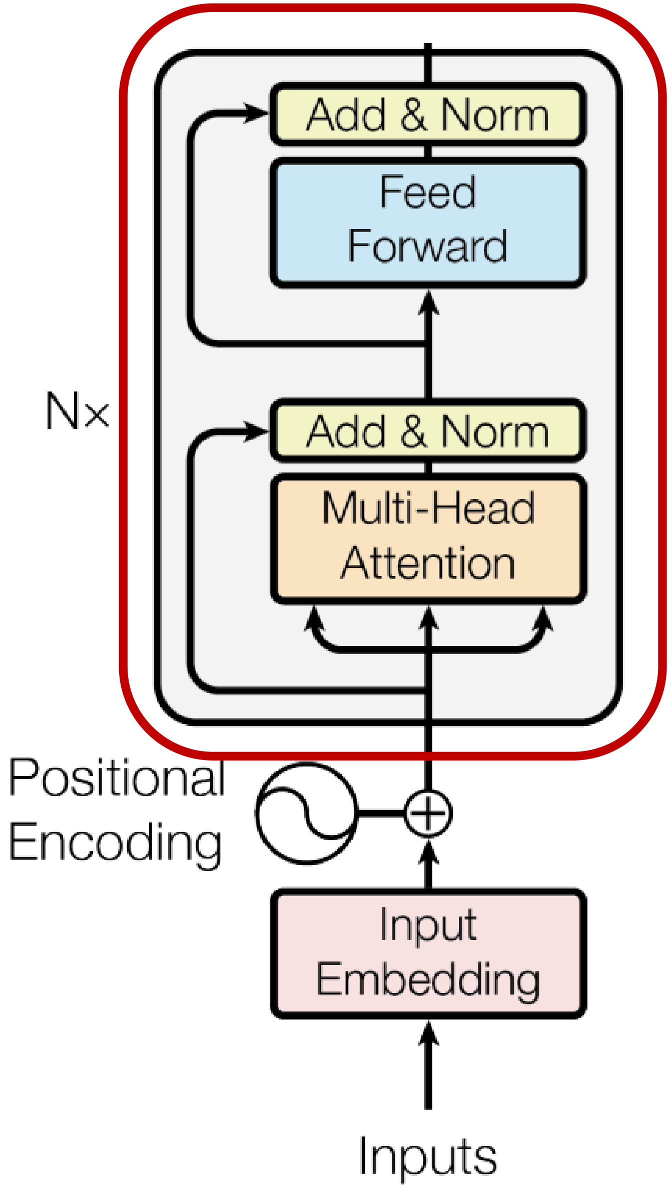
Transformer

- Encoder:
- Multi-Head Self-Attention
- Feed Forward
- Residual Connection & Layer Norm



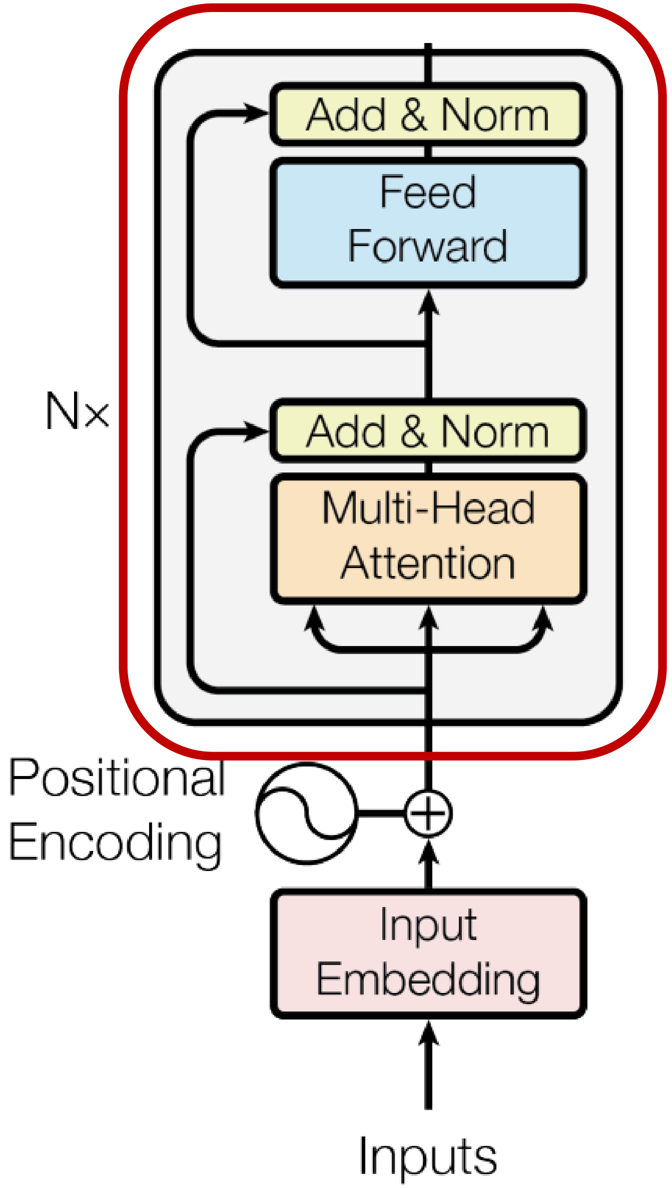
Transformer

$$\max(0, XW_1 + b_1)W_2 + b_2$$



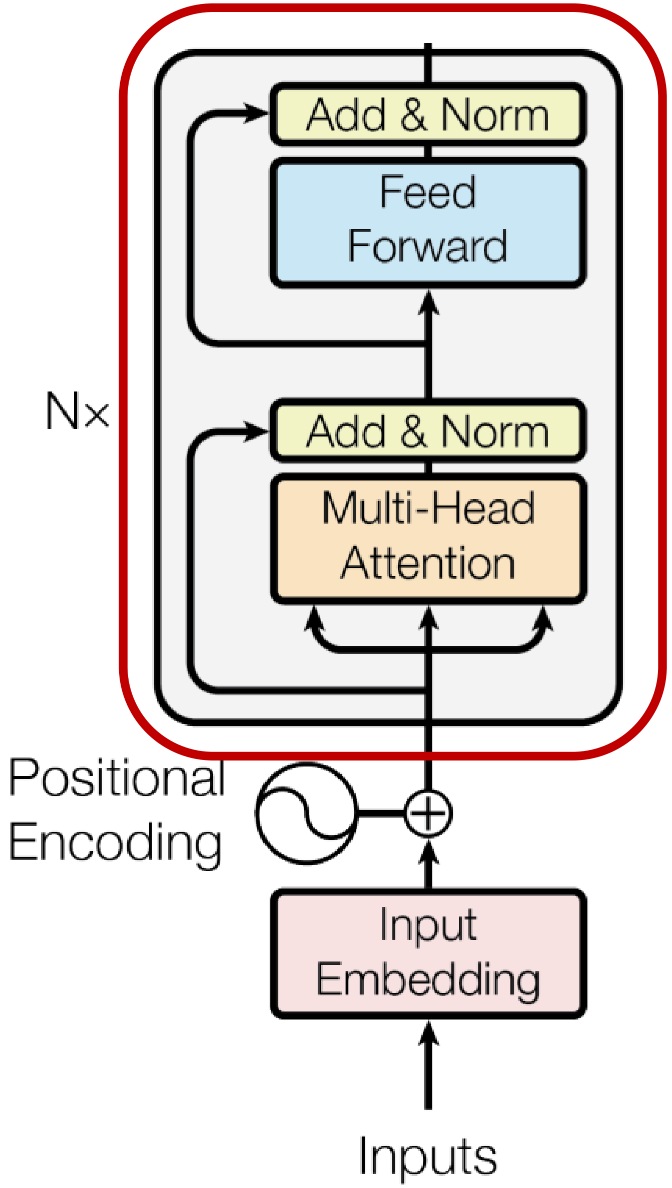
Transformer

- Encoder:
- Multi-Head Self-Attention
- Feed Forward
- Residual Connection & Layer Norm



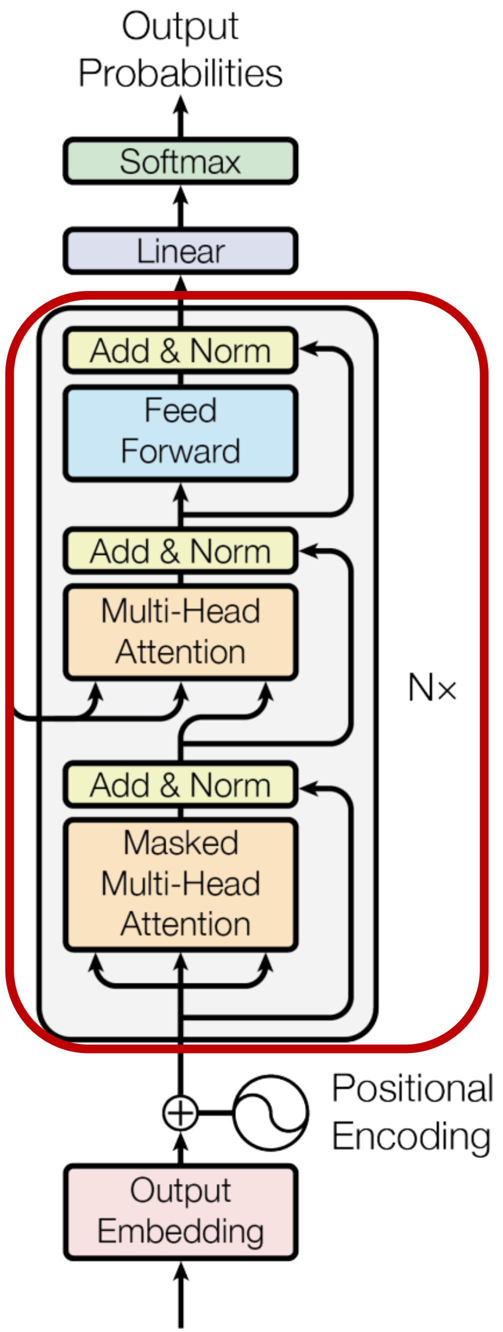
Add & Norm

$$\mathrm{LayerNorm}\big(X + \mathrm{MultiHeadAttention}(X)\big)$$
$$\mathrm{LayerNorm}\big(X + \mathrm{FeedForward}(X)\big)$$



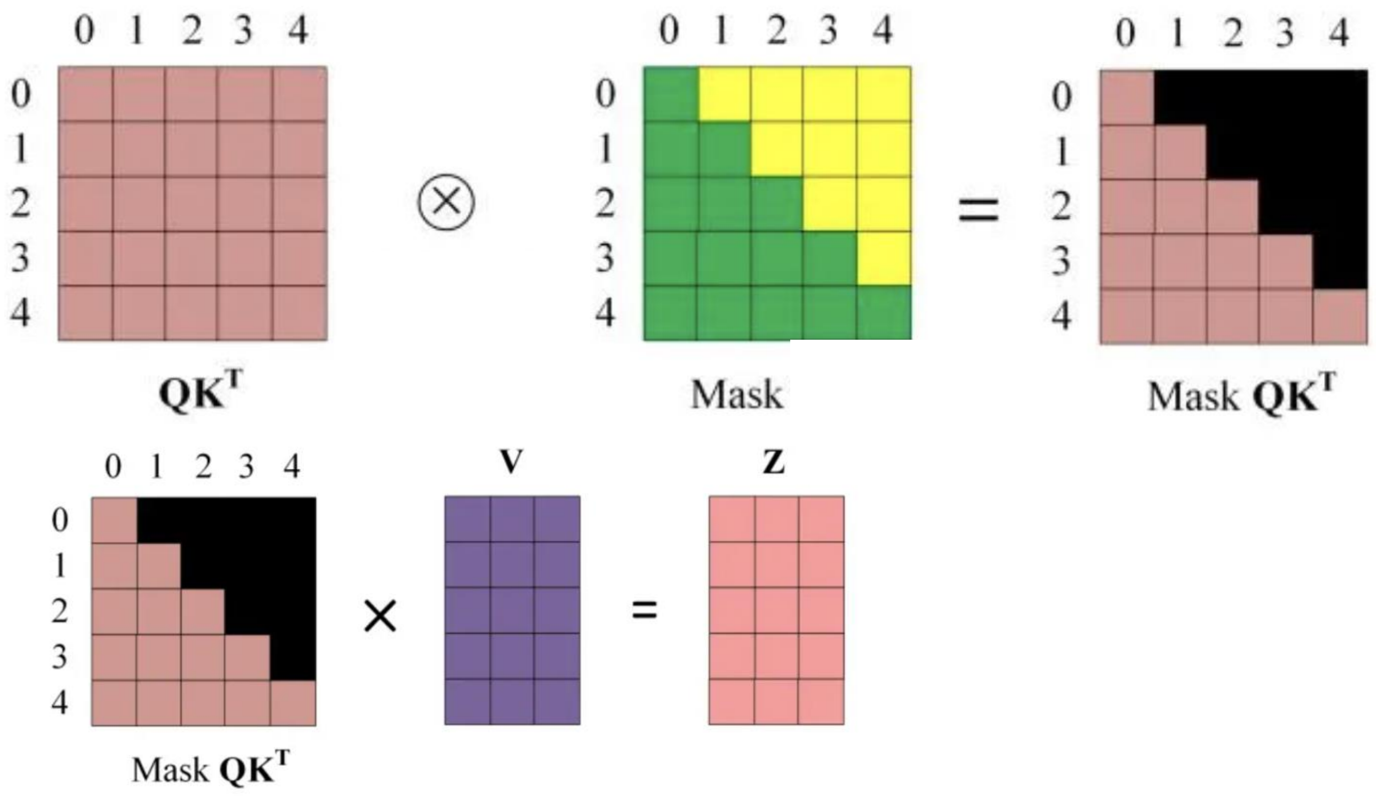
Transformer

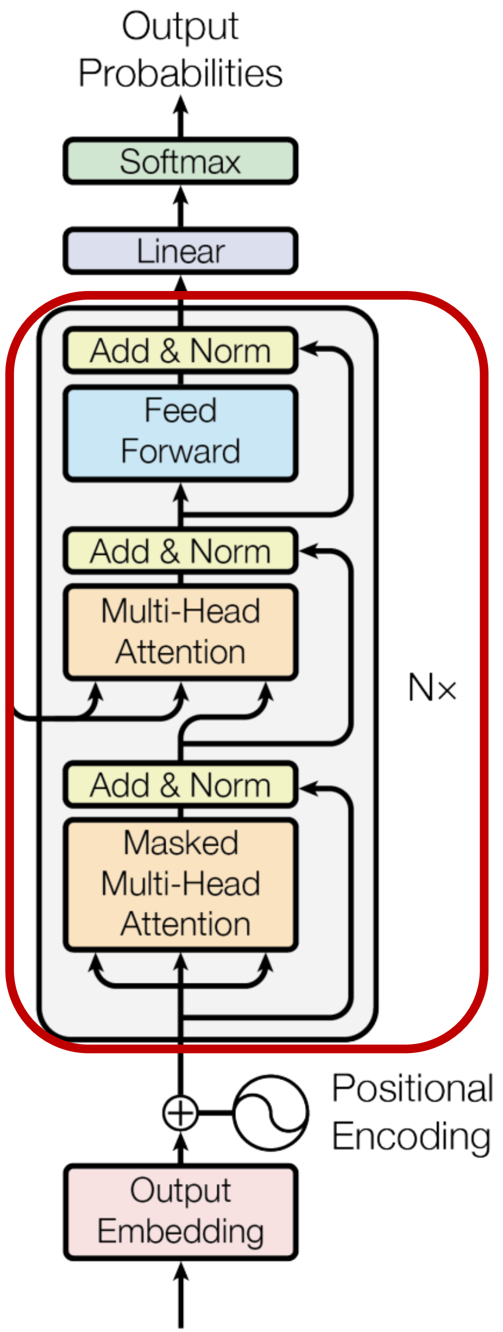
- Decoder:
- 2 x Multi-Head Self-Attention
- 1st Masked Multi-Head Self-Attention
- 2nd Q K V
- Softmax



Transformer

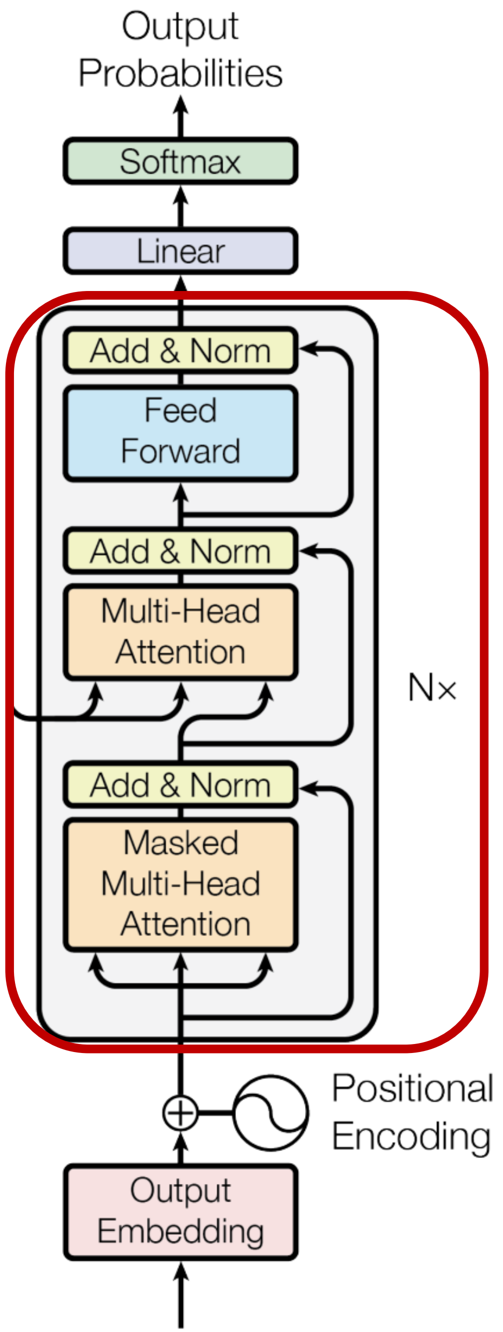
- Masked Multi-Head Self-Attention





Transformer

- Softmax

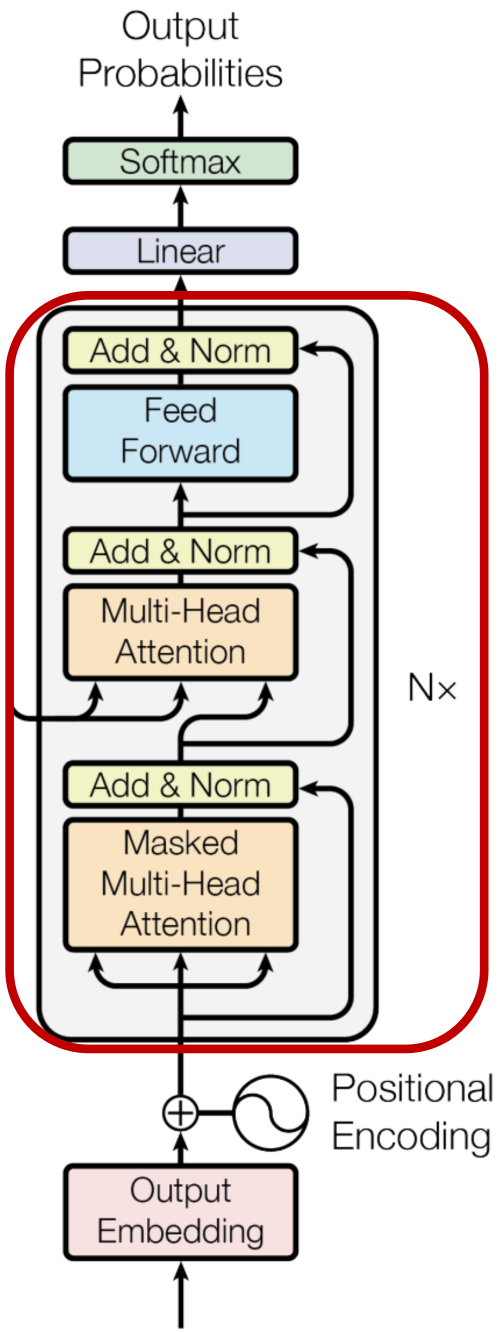


Transformer

- Positional Encoding

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

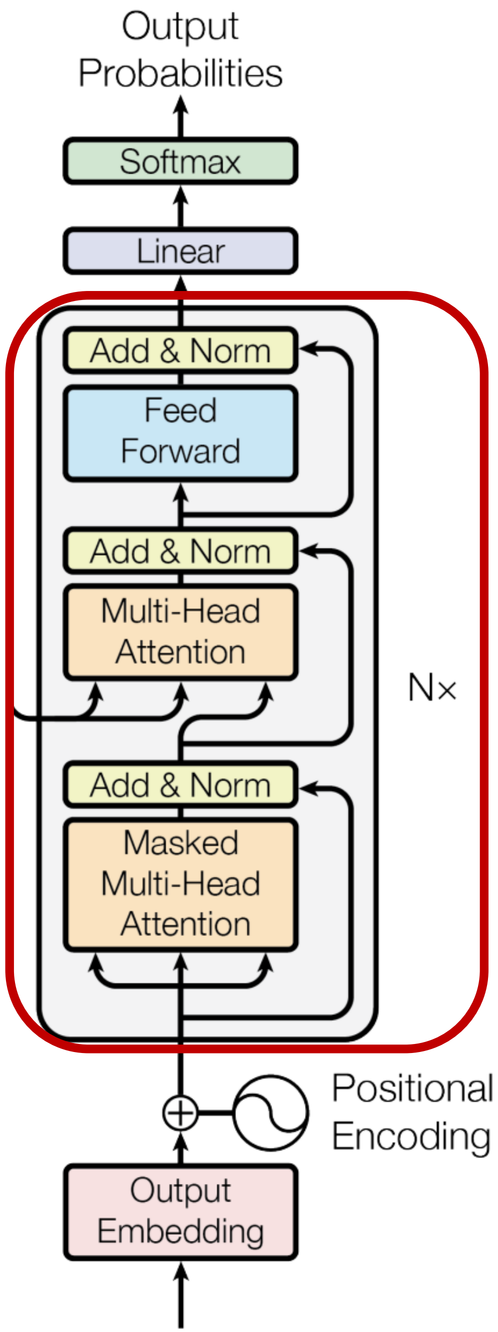


Transformer

- Positional Encoding

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1} \qquad M . \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t + \phi)) \\ \cos(\omega_k . (t + \phi)) \end{bmatrix}$$



Advantages of Transformer

- Enhanced Parallelization Capabilities
- Capturing Long-Distance Dependencies
- Dynamic Weight Allocation

Disadvantages of Transformer

- Computational Efficiency Issues
  - Quadratic Time Complexity
  - High Memory Consumption
- Limited Capability with Long Sequences
  - Limited Effective Resolution Window
  - Extended Training Times
- Overparameterization

SSMs——S4

Efficiently Modeling Long Sequences with Structured State Spaces

Albert Gu, Karan Goel, and Christopher Ré

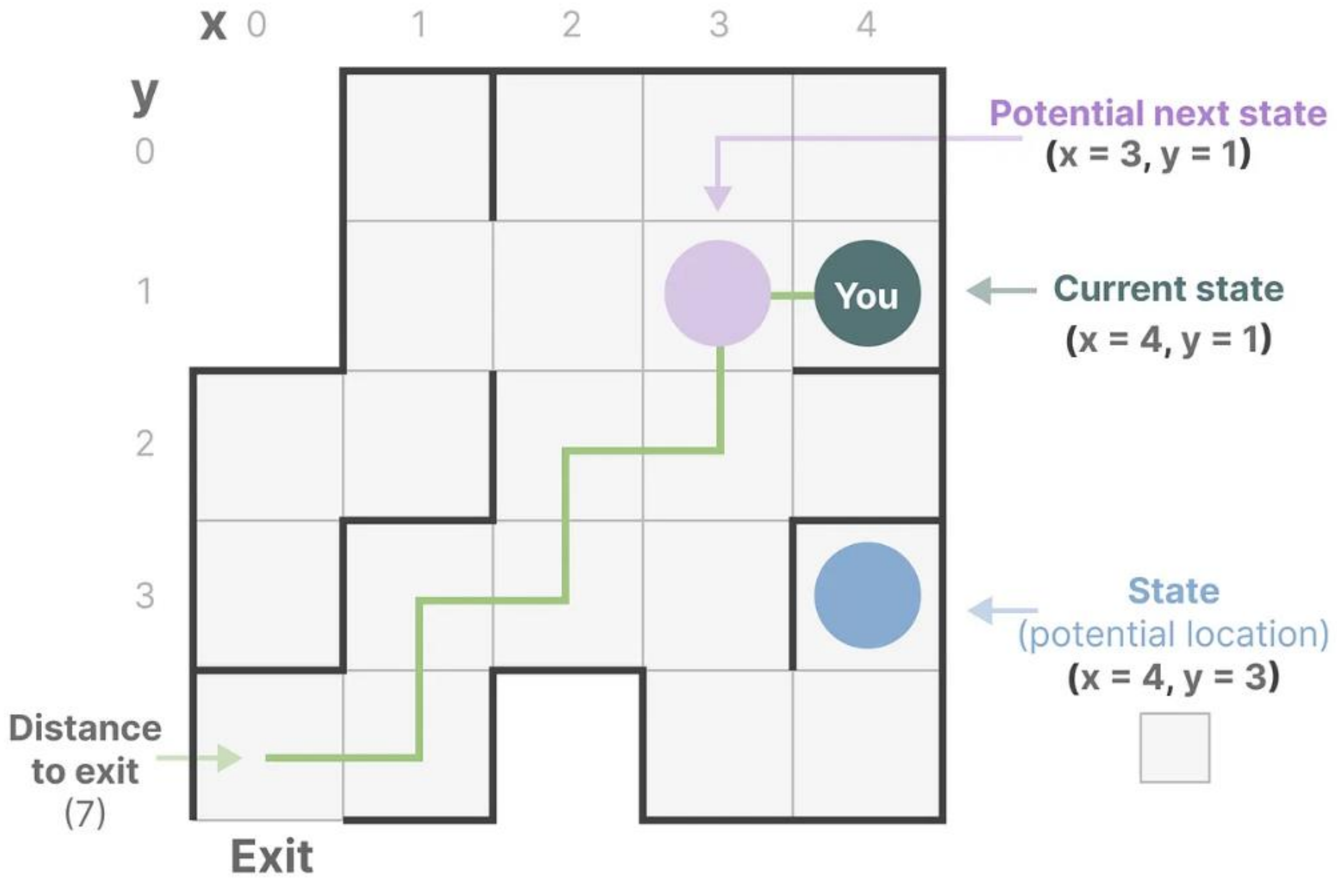
Department of Computer Science, Stanford University

{albertgu,krng}@stanford.edu, chrismre@cs.stanford.edu

Maarten Grootendorst《A Visual Guide to Mamba and State Space Models》

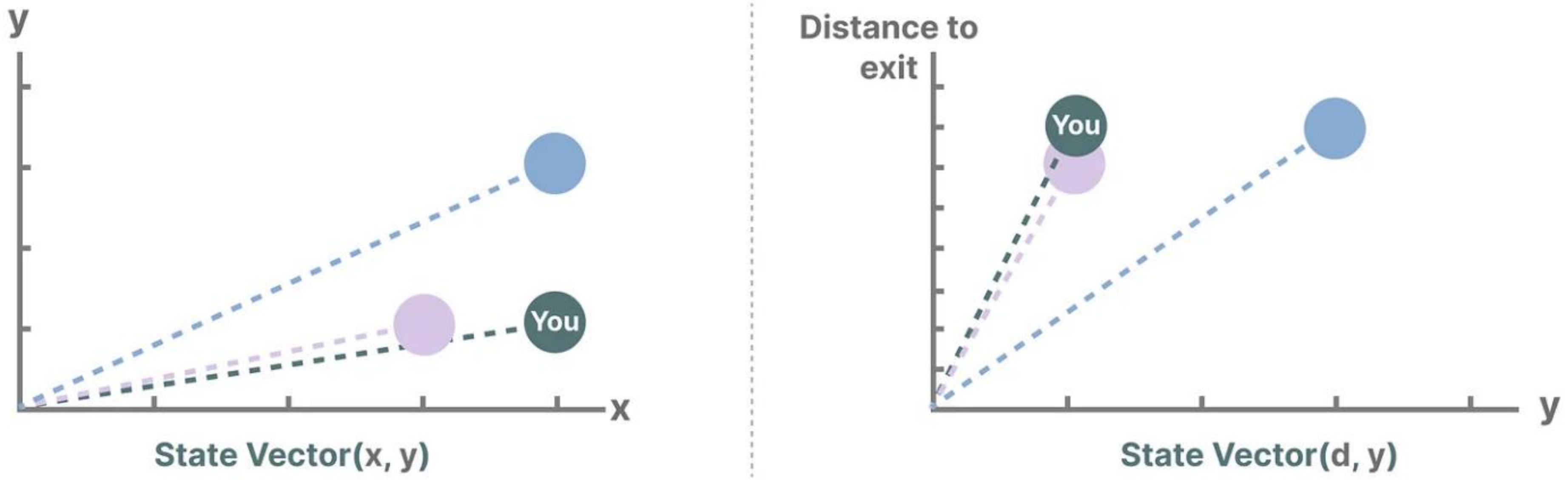
SSMs——S4

- State Space



SSMs——S4

- State Space

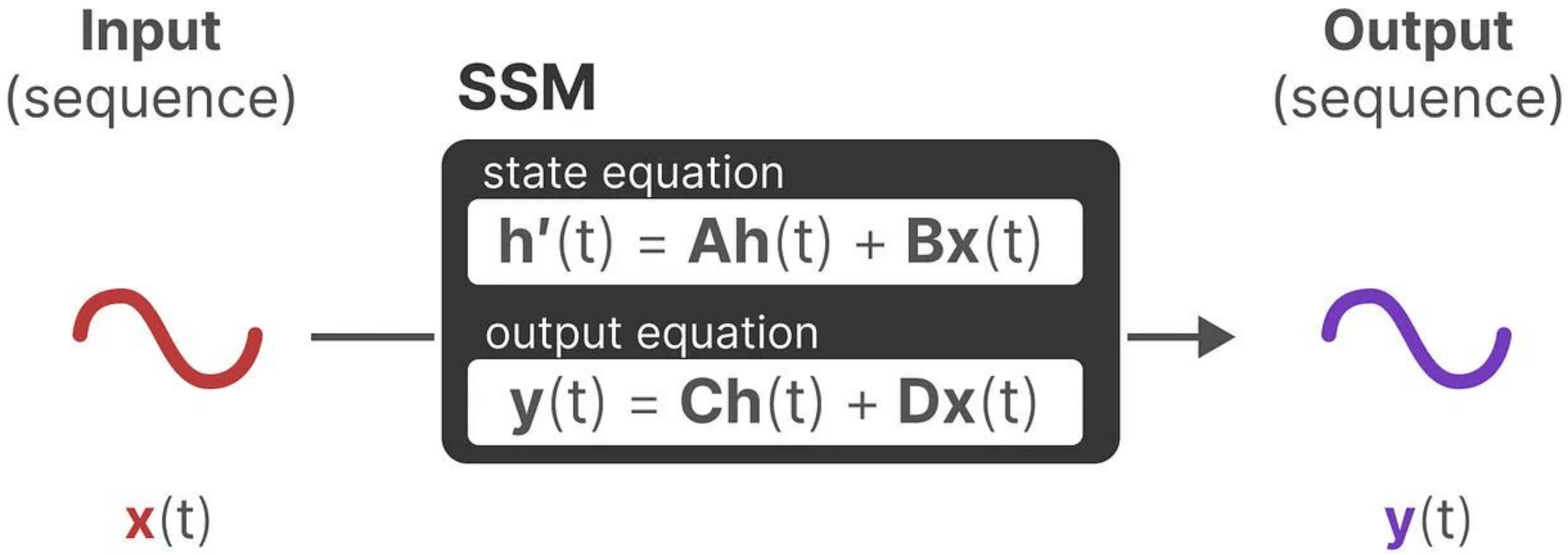


SSMs——S4

- SSM

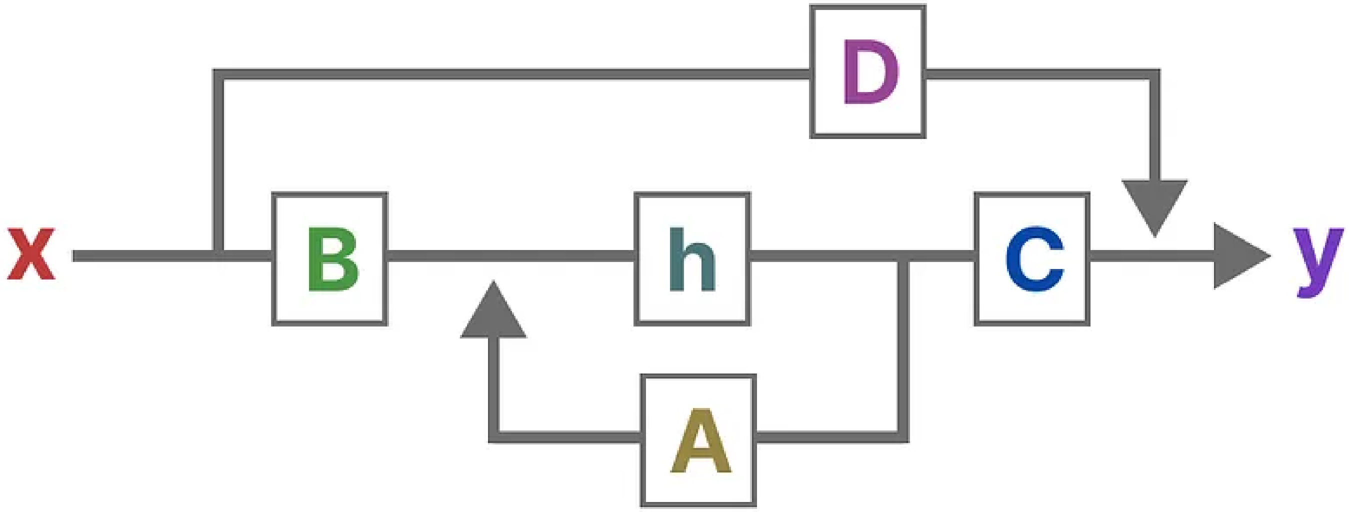
SSMs——S4

- SSM


Input
(sequence)
SSM
state equation
h'(t) = Ah(t) + Bx(t)
output equation
y(t) = Ch(t) + Dx(t)
Output
(sequence)
x(t)
y(t)

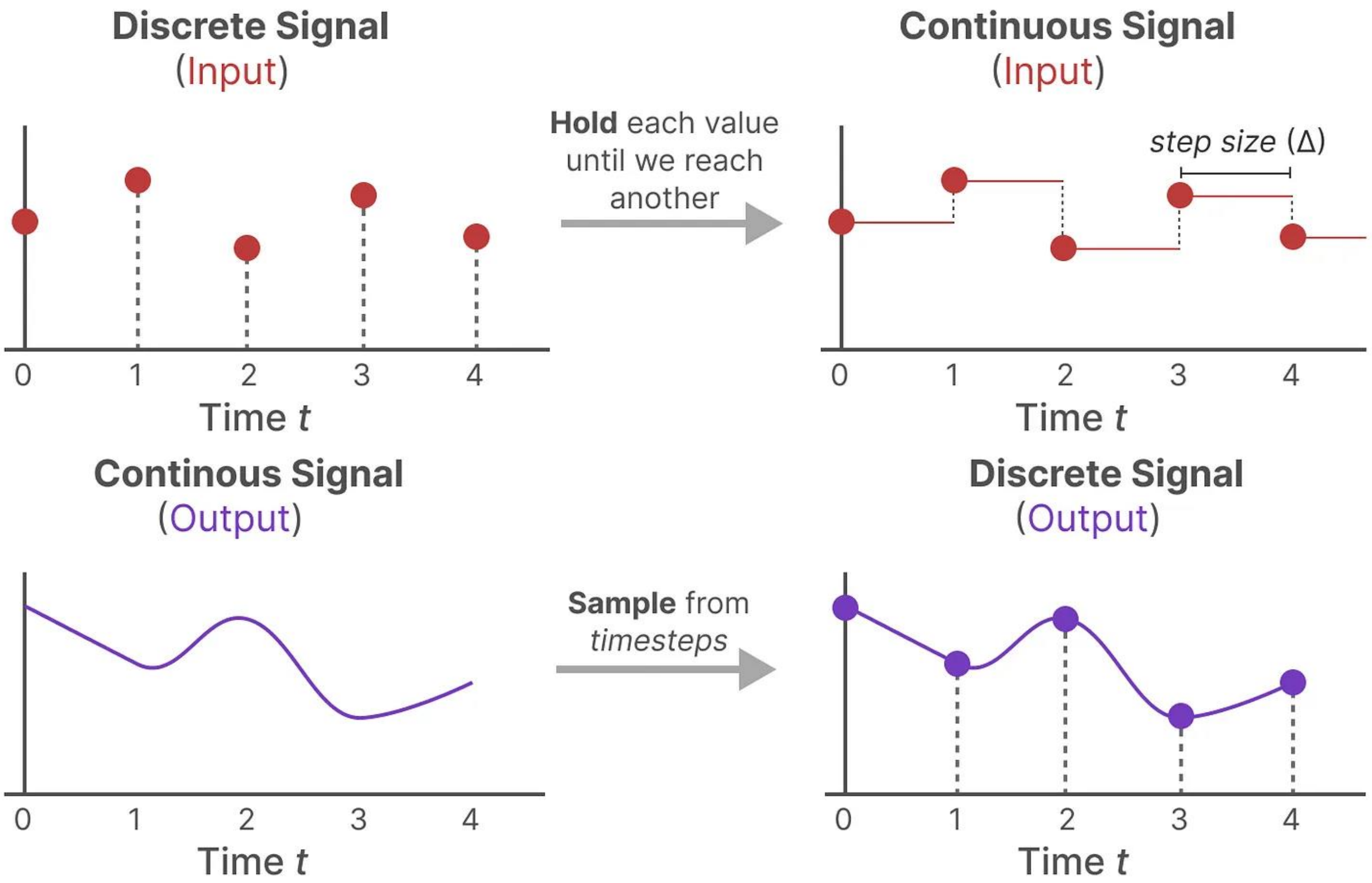
SSMs——S4

- SSM



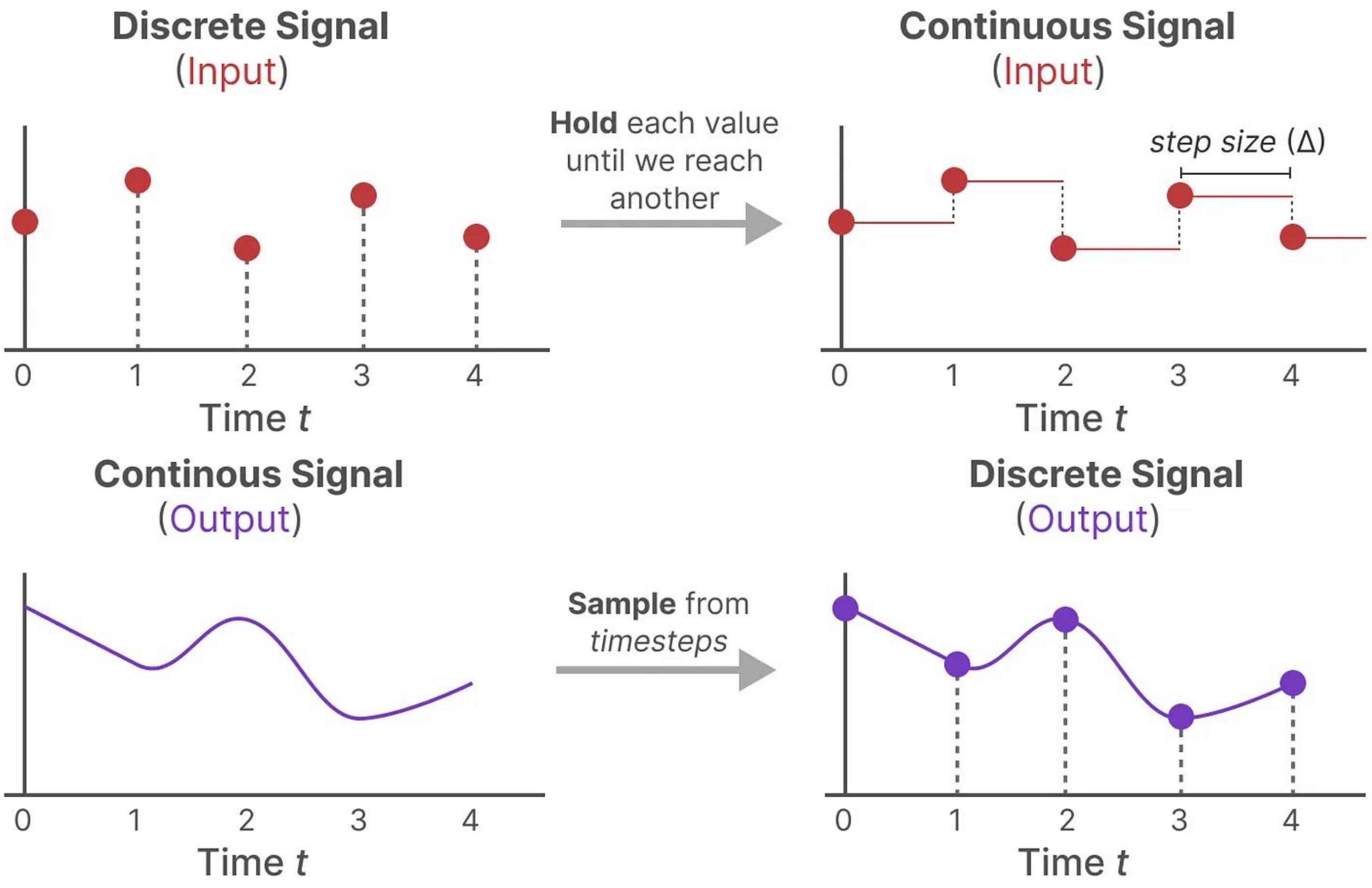
SSMs——S4

- S4


Discrete Signal
(Input)
0 1 2 3 4
Time t
Hold each value until we reach another
Continuous Signal
(Input)
step size (Δ)
0 1 2 3 4
Time t
Continous Signal
(Output)
0 1 2 3 4
Time t
Sample from timesteps
Discrete Signal
(Output)
0 1 2 3 4
Time t

SSMs——S4

- S4



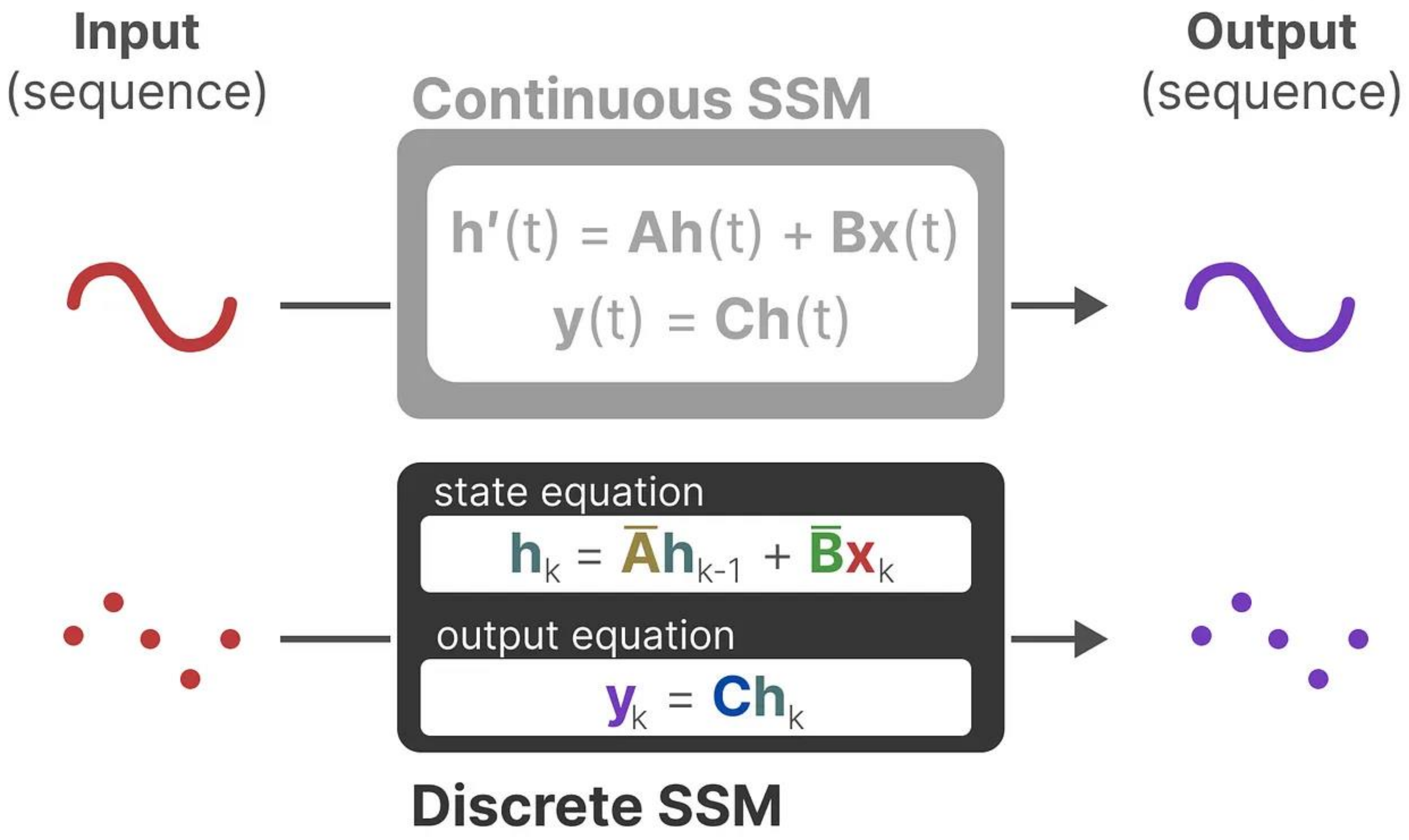
Discrete Signal
(Input)
0
1
2
3
4
Time t
Hold each value
until we reach
another
Continuous Signal
(Input)
step size (Δ)
0
1
2
3
4
Time t
Continous Signal
(Output)
0
1
2
3
4
Time t
Sample from
timesteps
Discrete Signal
(Output)
0
1
2
3
4
Time t

SSMs——S4

- S4

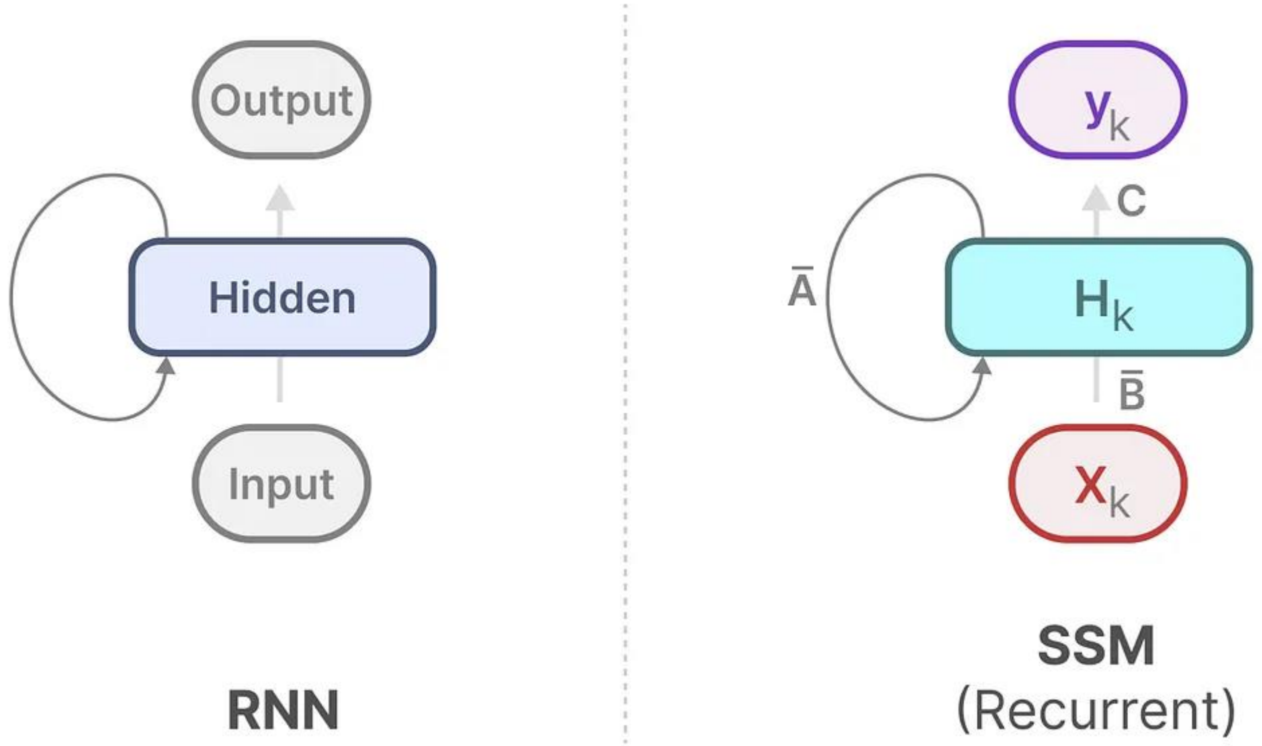
Discretized matrix **A** $\bar{\mathbf{A}} = \exp(\Delta \mathbf{A})$

Discretized matrix **B** $\bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - I) \cdot \Delta \mathbf{B}$



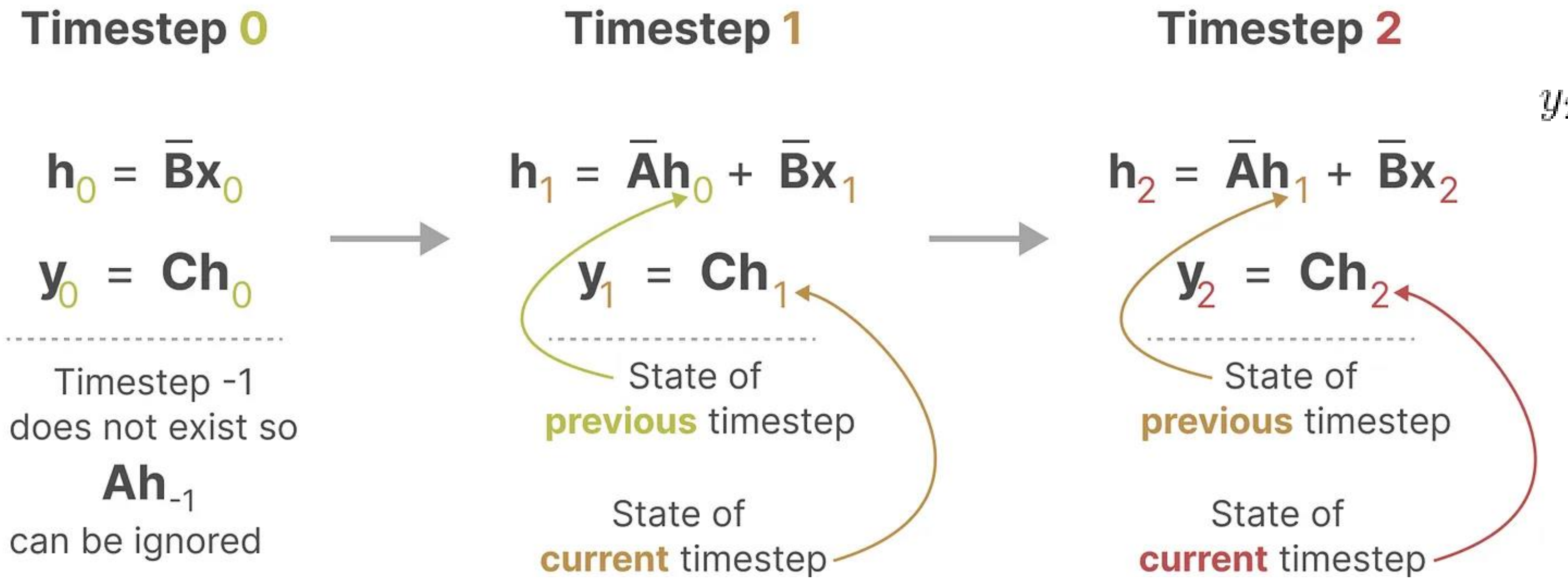
SSMs——S4

- S4



SSMs——S4

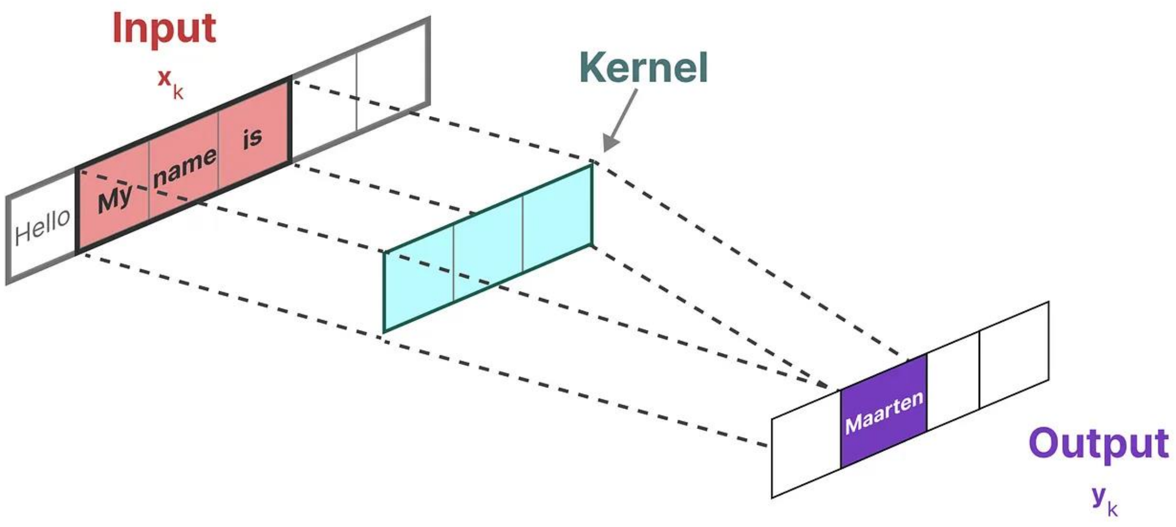
- S4



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

SSMs——S4

- S4



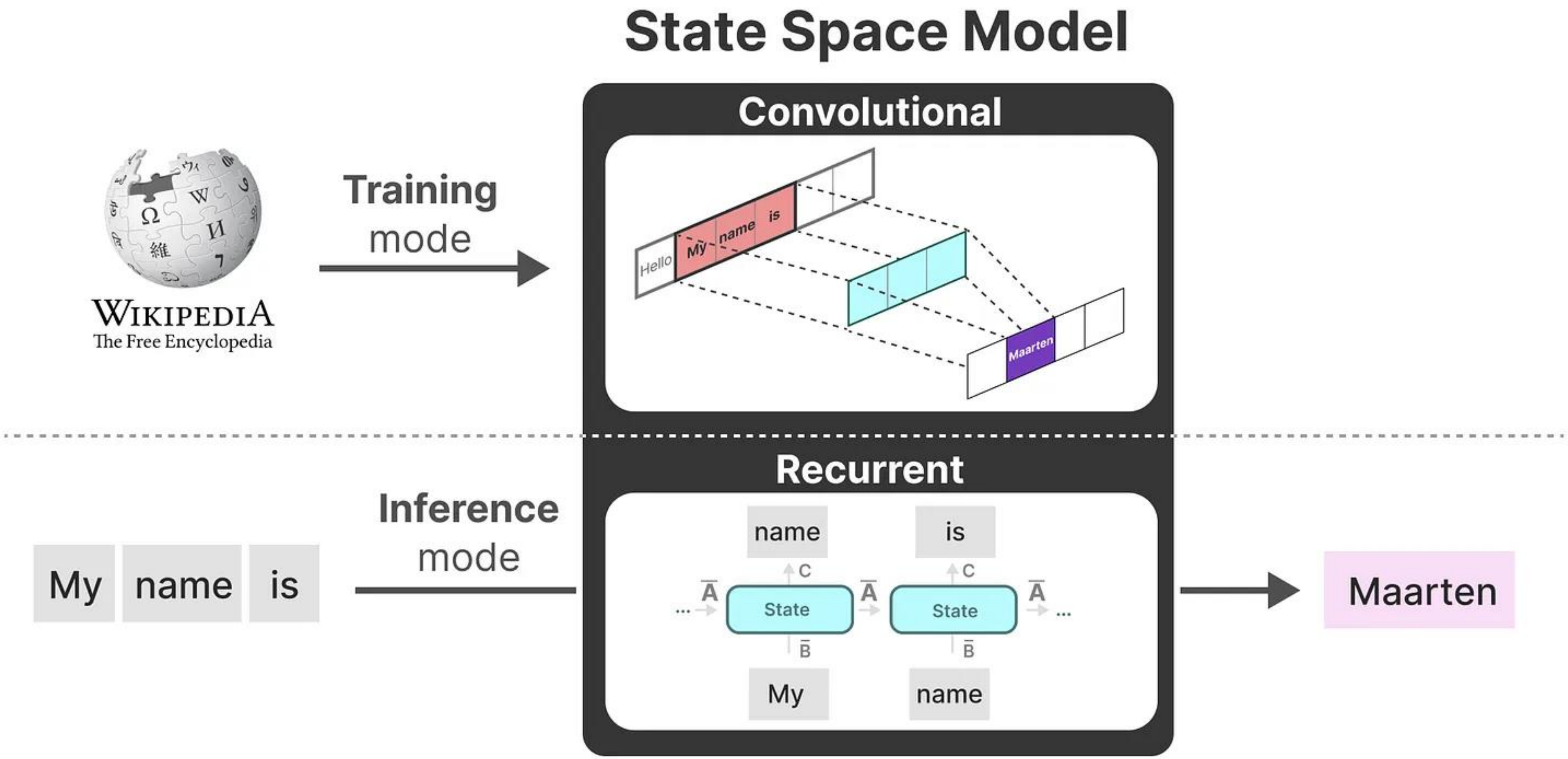
kernel $\longrightarrow$ $\overline{K} = (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^{k}\overline{B}, \dots)$

$$\underset{\text{output}}{y} = \underset{\text{input}}{x} * \underset{\text{kernel}}{\overline{K}}$$

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

SSMs——S4

- S4



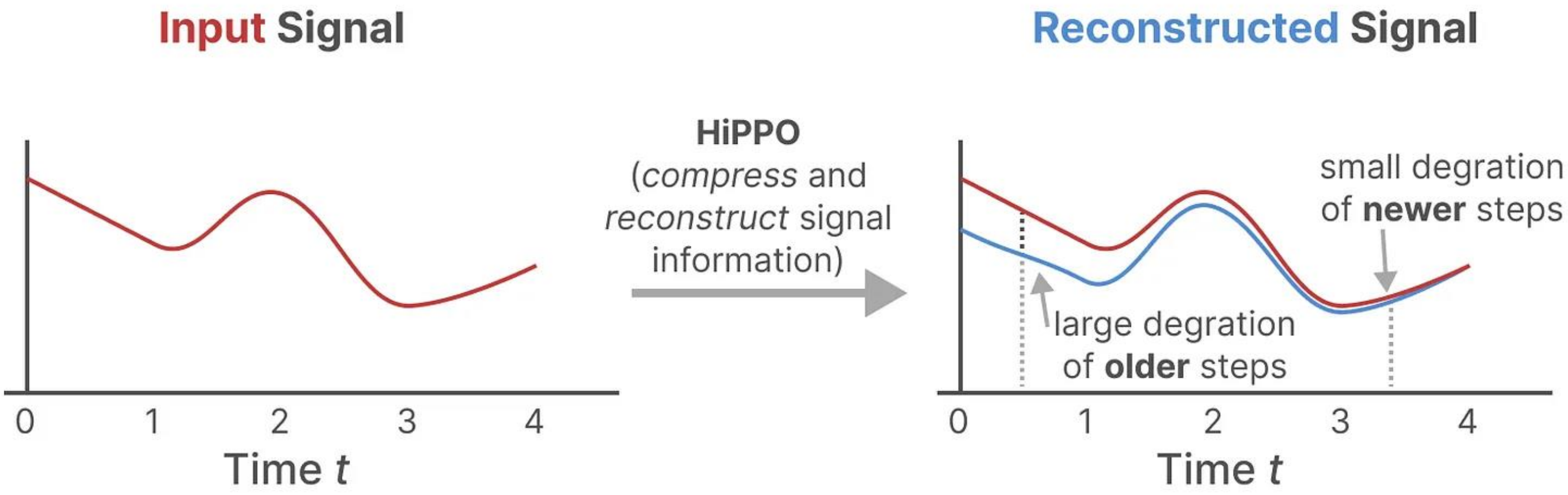
State Space Model
Convolutional
Training
mode
WIKIPEDIA
The Free Encyclopedia
Hello
My
name
is
Maarten
Recurrent
Inference
mode
My
name
is
name
is
C
C
Ā
Ā
Ā
State
State
B̄
B̄
My
name
...
...
Maarten

SSMs——S4

HiPPO: Recurrent Memory with Optimal Polynomial Projections

Albert Gu[*†], **Tri Dao**[*†], **Stefano Ermon** [†], **Atri Rudra** [‡], **Christopher Ré** [†]
[†] Department of Computer Science, Stanford University
[‡] Department of Computer Science and Engineering, University at Buffalo, SUNY
{albertgu,trid}@stanford.edu, ermon@cs.stanford.edu, atri@buffalo.edu, chrismre@cs.stanford.edu

SSMs——S4


Input Signal
Reconstructed Signal
HiPPO
(*compress* and *reconstruct* signal information)
small degration of **newer** steps
large degration of **older** steps
0
1
2
3
4
Time *t*
0
1
2
3
4
Time *t*

SSMs——S4

HiPPO Matrix

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

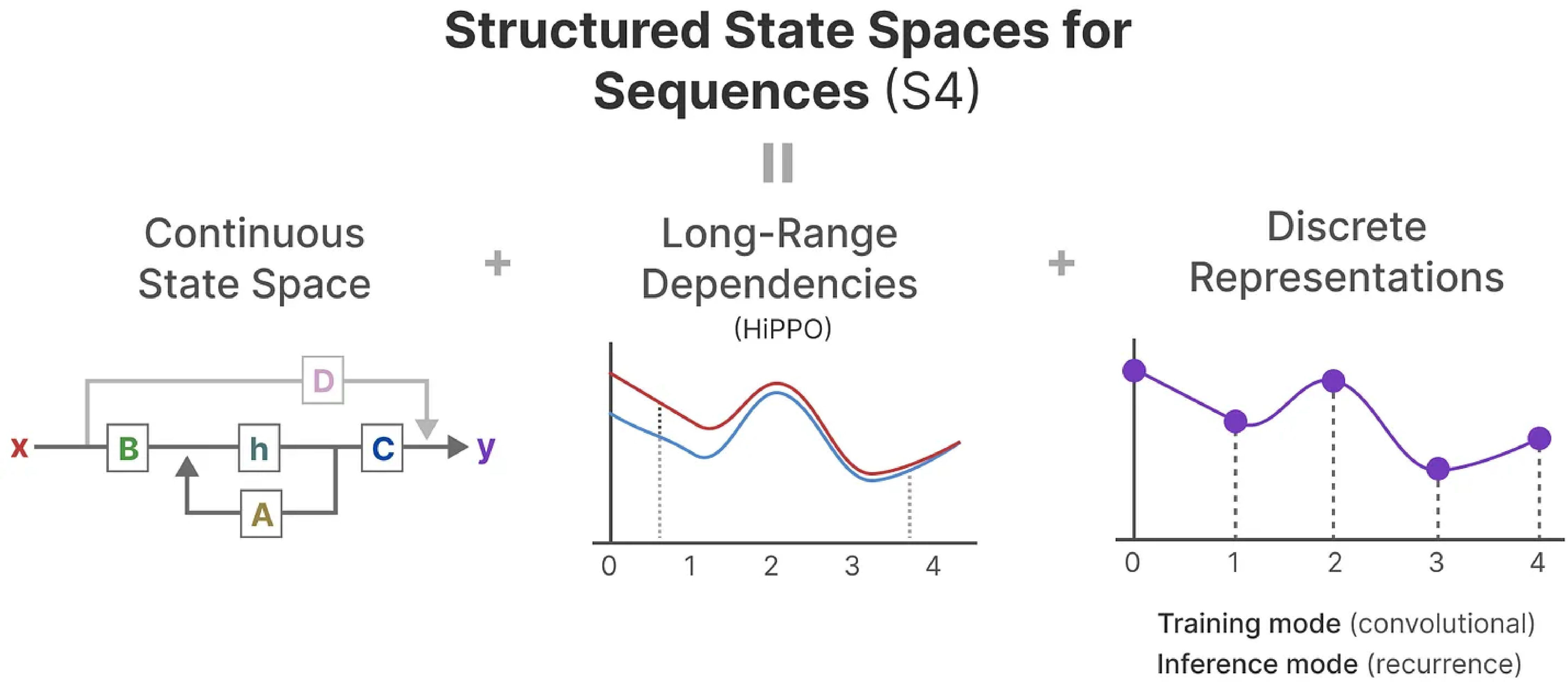
HiPPO Matrix

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

k

n

SSMs——S4


Structured State Spaces for Sequences (S4)
Continuous State Space
+
Long-Range Dependencies
(HiPPO)
+
Discrete Representations
x
B
h
C
y
D
A
0
1
2
3
4
0
1
2
3
4
Training mode (convolutional)
Inference mode (recurrence)

Mamba——S6

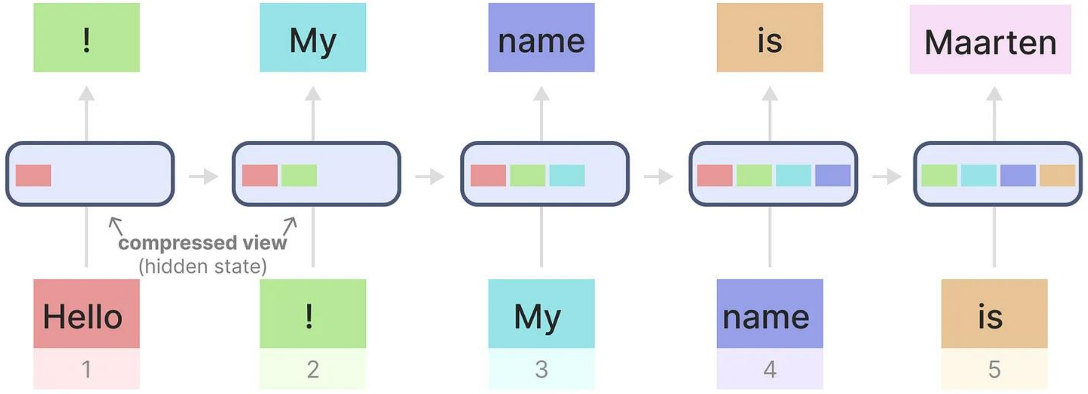
- Selection Mechanism

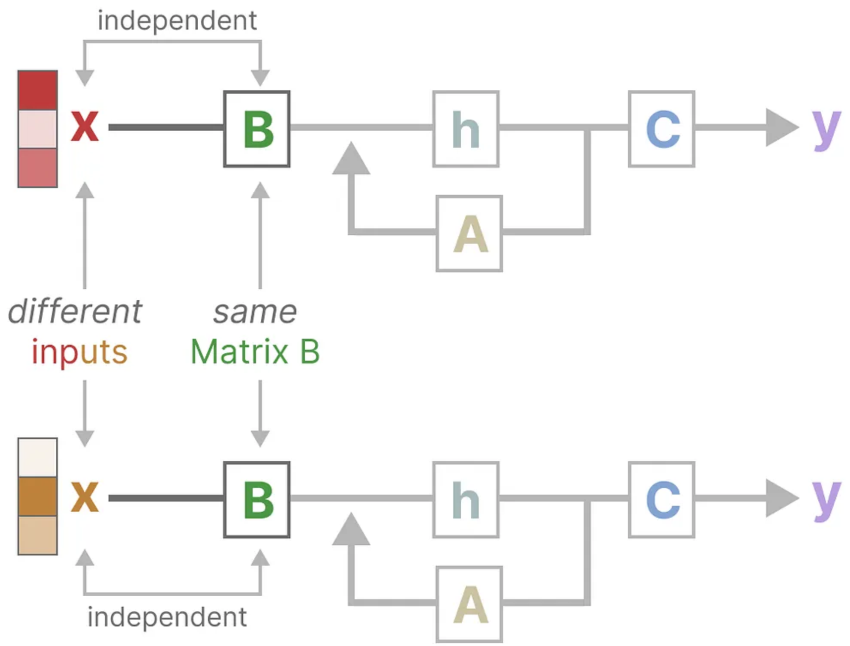
- Hardware-aware Algorithm

- Simpler SSM Architecture

Disadvantages of Previous Works

- Transformer——long context
- RNN——forget far context
- S4——fixed A, B, C





Mamba——S6

- Selection Mechanism

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

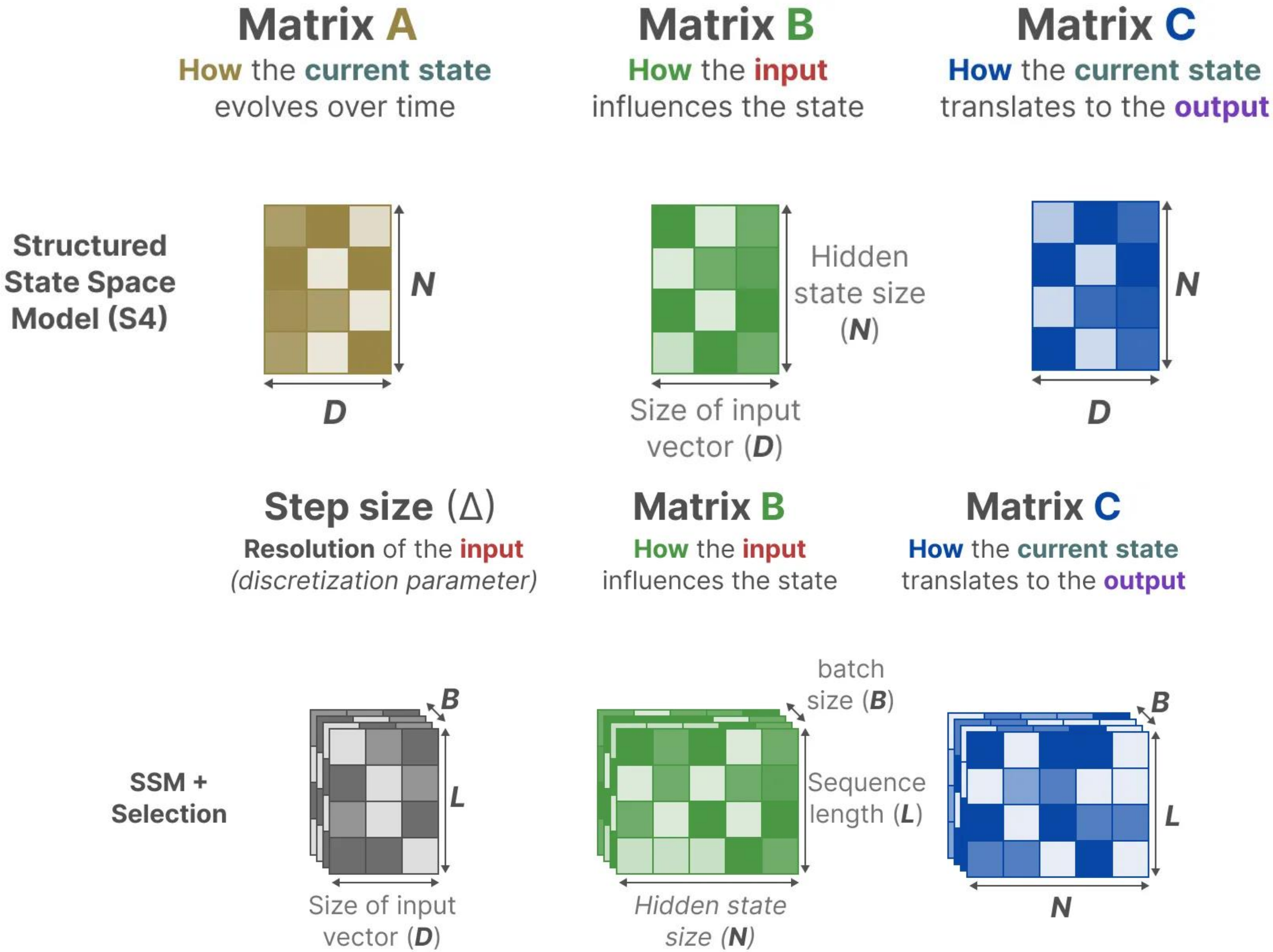
$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\triangleright$ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D})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

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

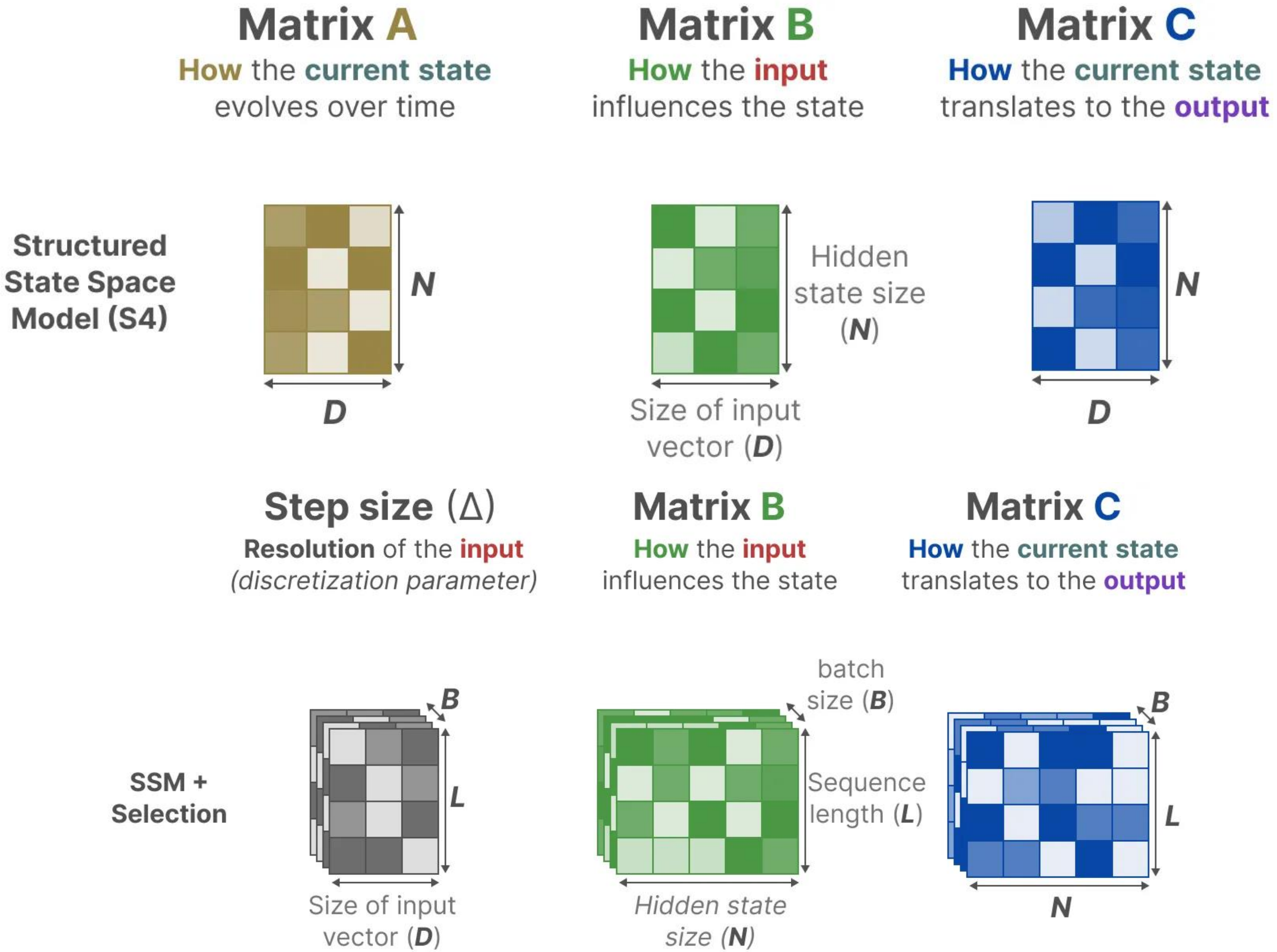
$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\triangleright$ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
$\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
$\Delta : (\mathtt{B},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

Mamba——S6



$$s_B(x) = \mathrm{Linear}_N(x)$$
$$s_C(x) = \mathrm{Linear}_N(x)$$
$$s_\Delta(x) = \mathrm{Linear}_D(x)$$
$$\tau_\Delta = \mathrm{softplus}$$

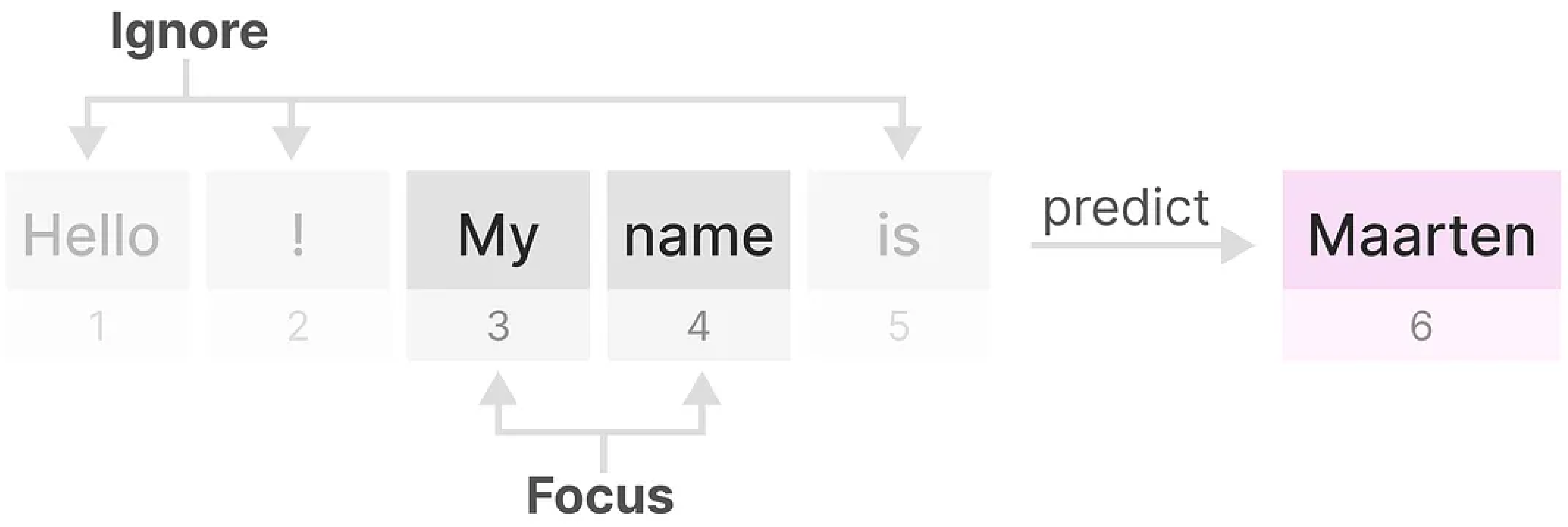
Mamba——S6



$$s_B(x) = \mathrm{Linear}_N(x)$$
$$s_C(x) = \mathrm{Linear}_N(x)$$
$$s_\Delta(x) = \mathrm{Linear}_D(x)$$
$$\tau_\Delta = \mathrm{softplus}$$

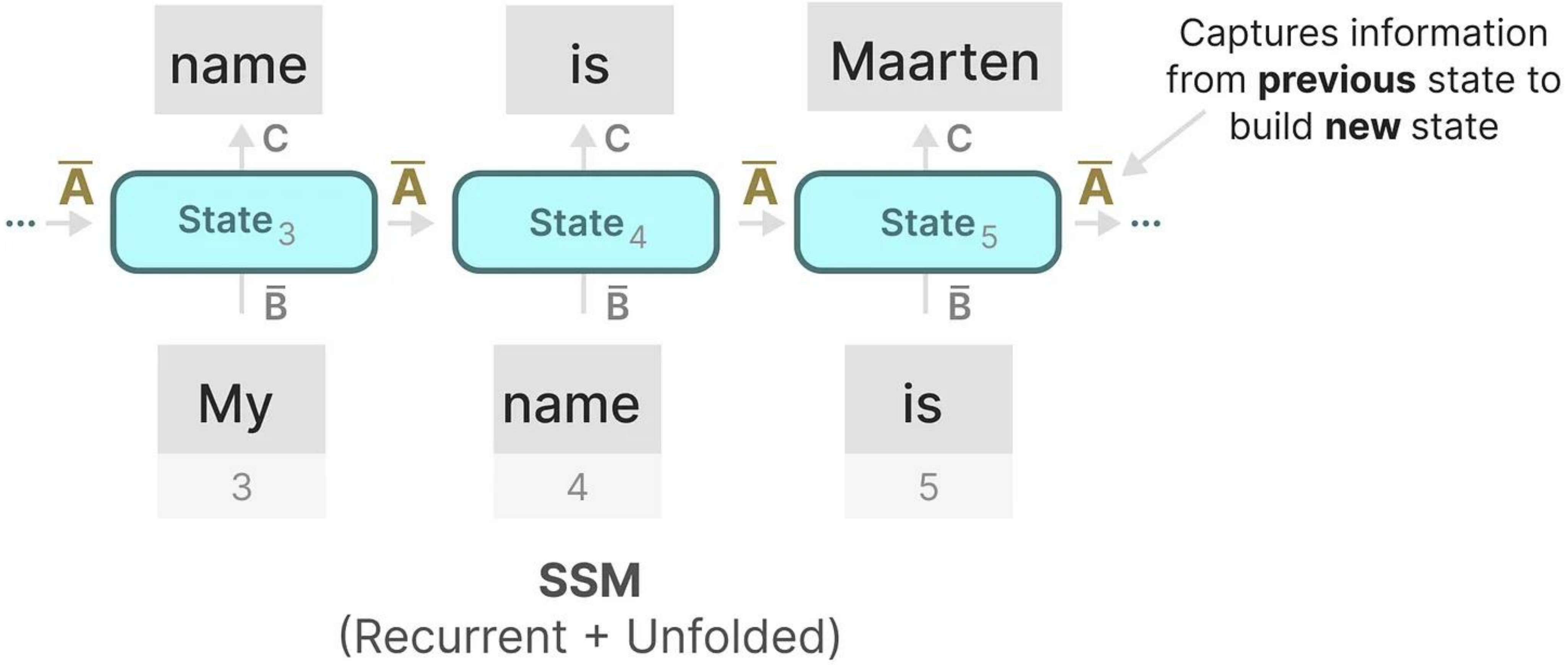
Mamba——S6

- Selection Mechanism



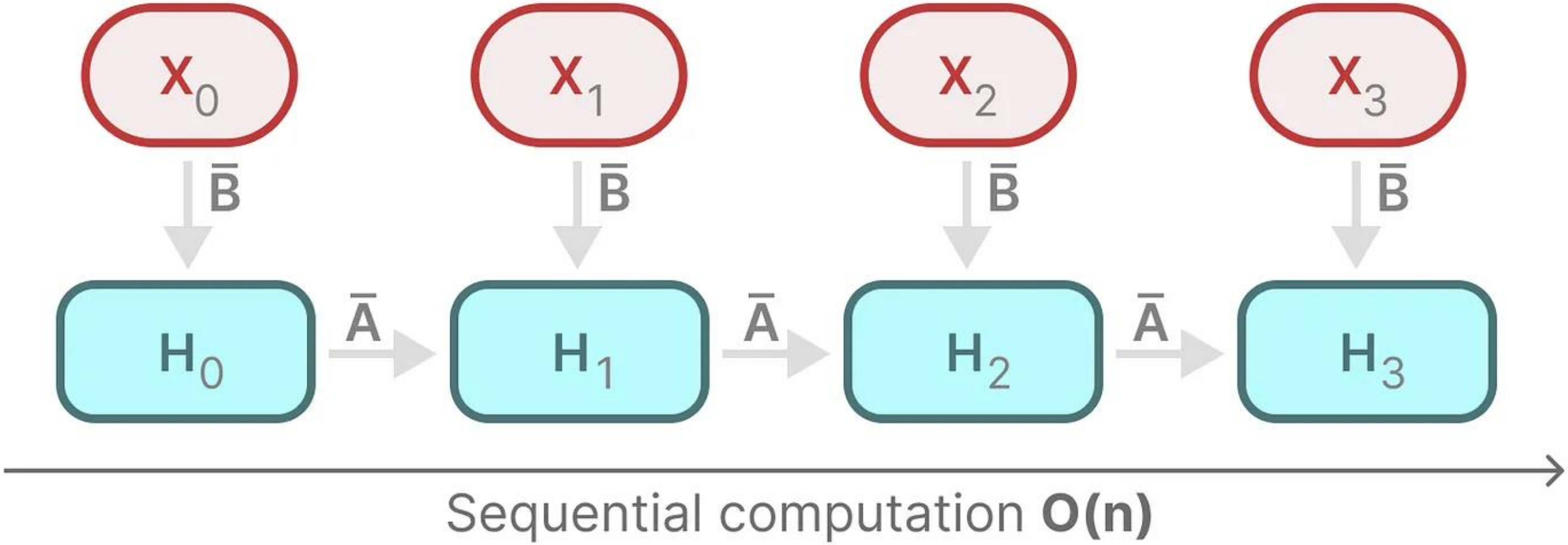
Mamba——S6

- Selection Mechanism



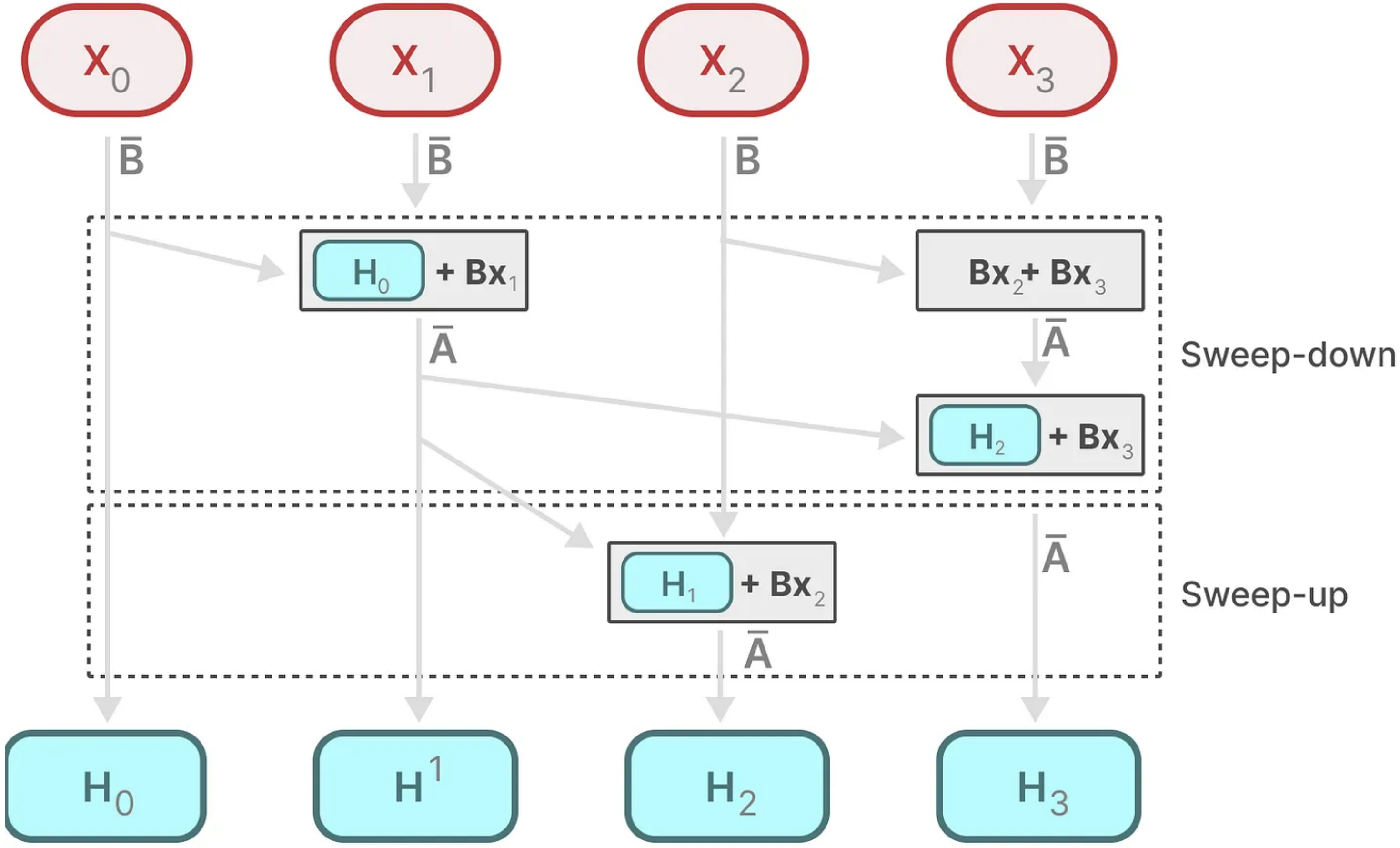
Mamba——S6

- Hardware-aware Algorithm



Mamba——S6

- Hardware-aware Algorithm

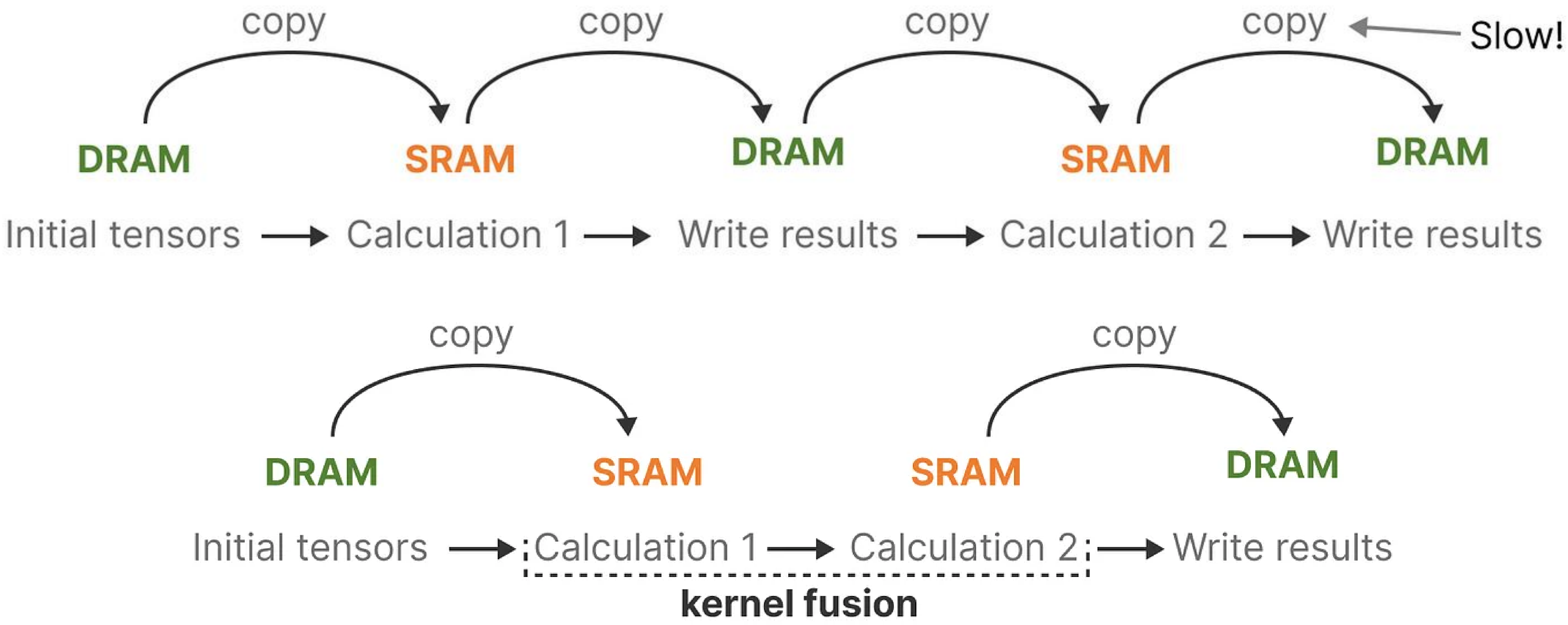


Parallel computation **O(n/t)**

$$
\begin{aligned}
H_1 &= \bar{A} \cdot H_0 + BX_1 \\
H_2 &= \bar{A} \cdot H_1 + BX_2 \\
&= \bar{A} \cdot \left(\bar{A} H_0 + BX_1\right) + BX_2 \\
H_3 &= \bar{A} \cdot H_2 + BX_3 \\
&= \bar{A} \cdot \left(\bar{A} \cdot H_1 + BX_2\right) + BX_3 \\
&= \bar{A} \cdot \left(\bar{A} \cdot \left(\bar{A} H_0 + BX_1\right) + BX_2\right) + BX_3
\end{aligned}
$$

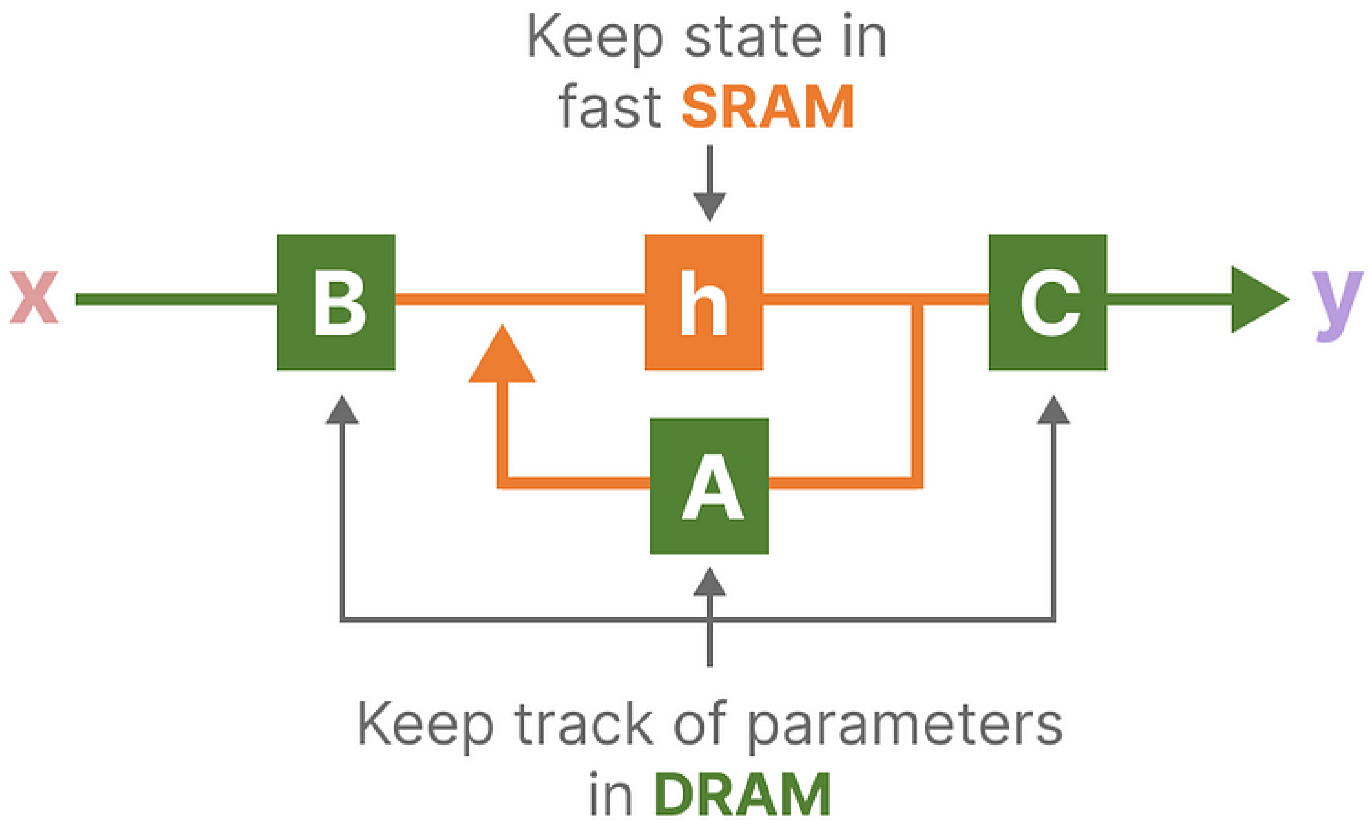
Mamba——S6

- Hardware-aware Algorithm (2)



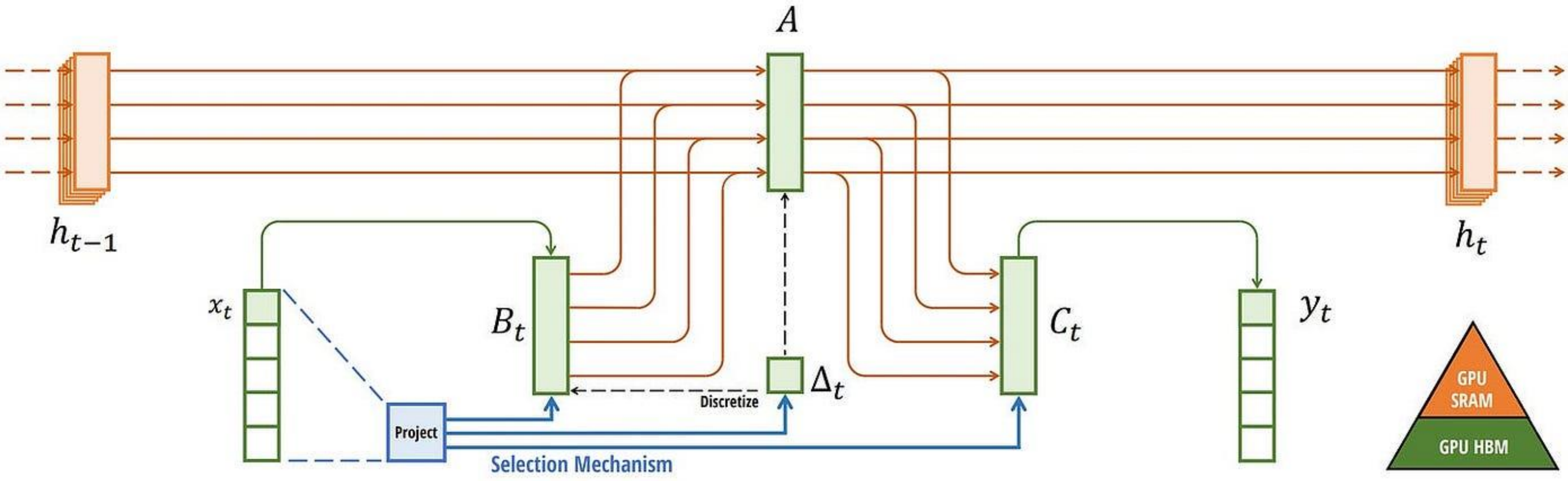
Mamba——S6

- Hardware-aware Algorithm (2)

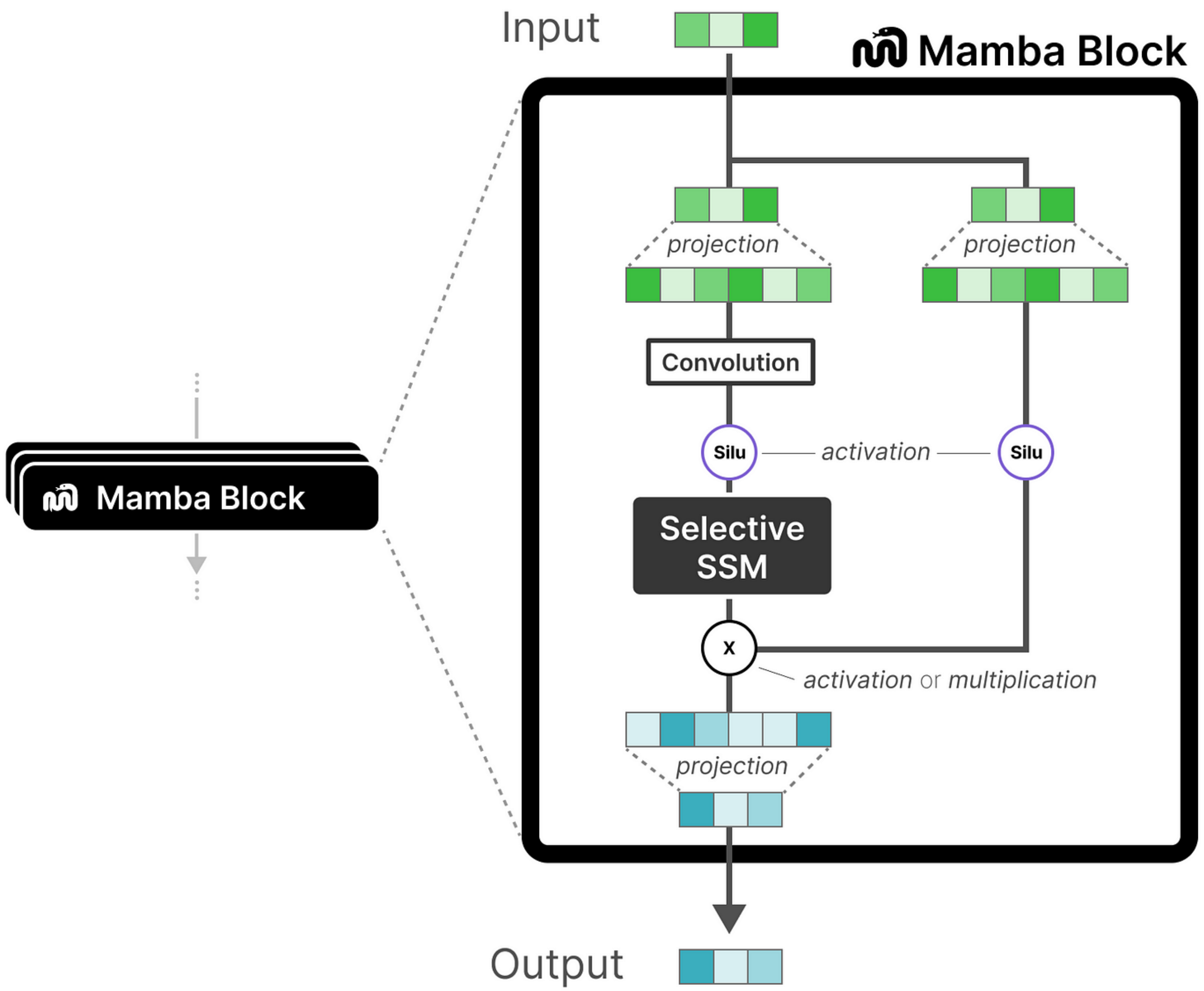


Mamba——S6

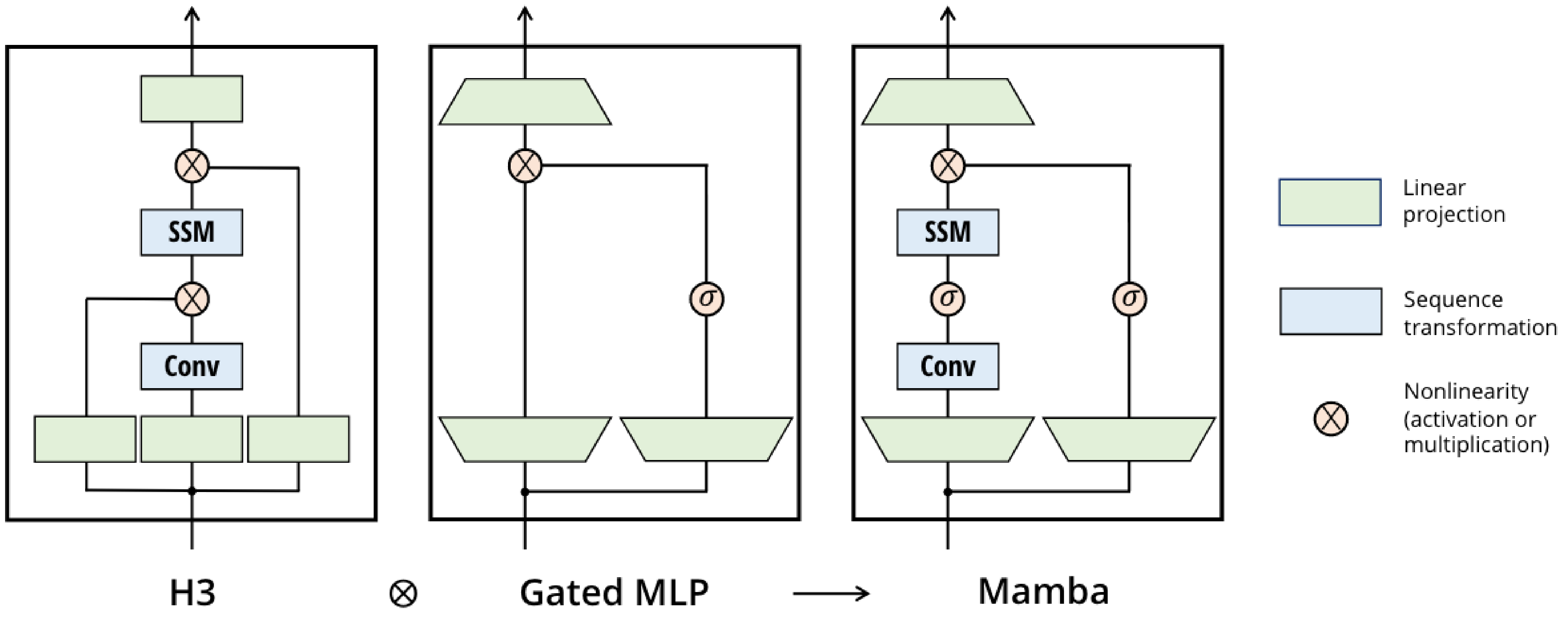
- Hardware-aware Algorithm (2)



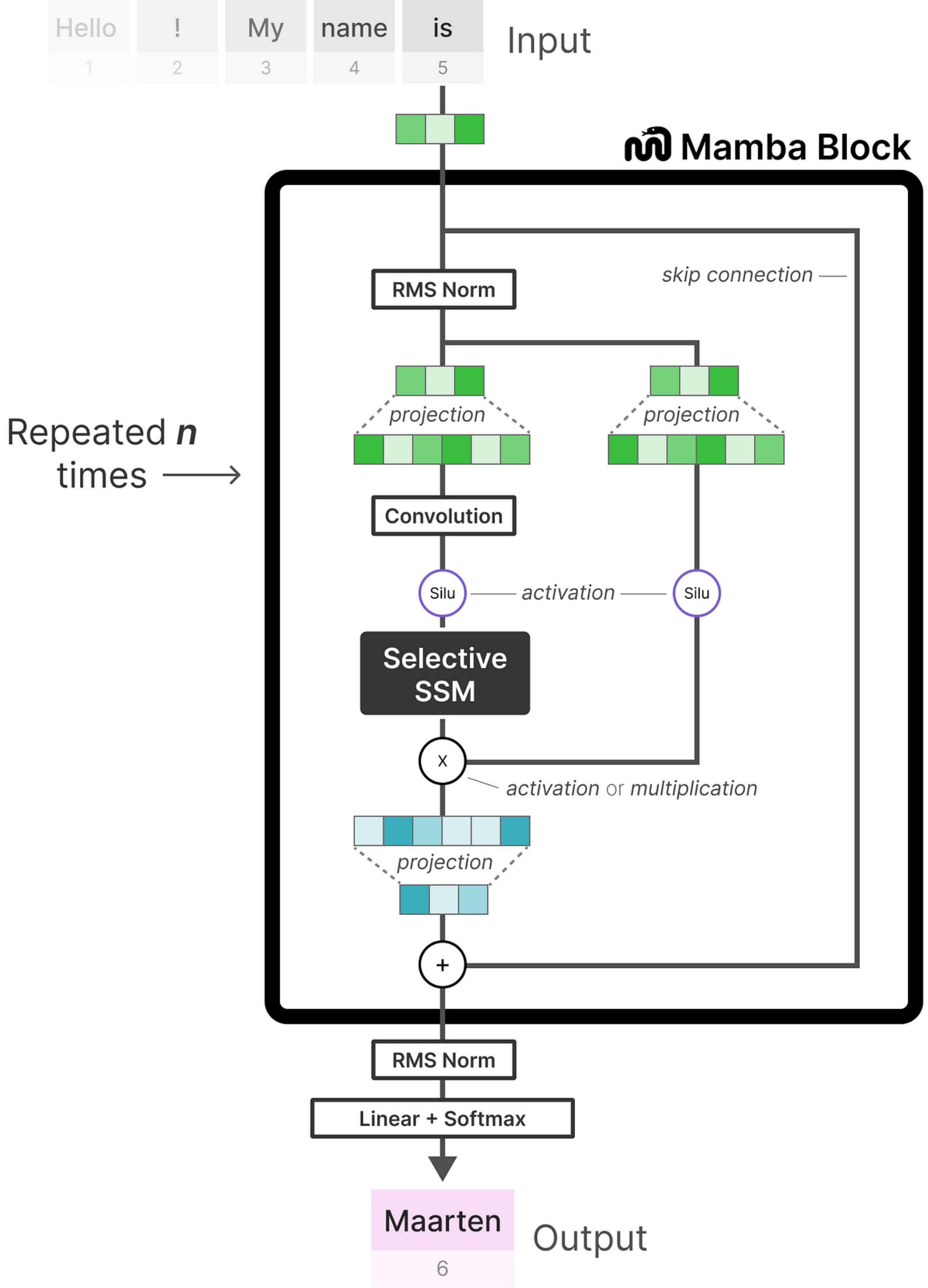
Mamba——S6


Input
Mamba Block
Mamba Block
projection
projection
Convolution
Silu
activation
Silu
Selective SSM
X
activation or multiplication
projection
Output

Mamba——S6


SSM
Conv
SSM
Conv
σ
σ
σ
Linear projection
Sequence transformation
Nonlinearity (activation or multiplication)
H3
⊗
Gated MLP
Mamba

Mamba——S6

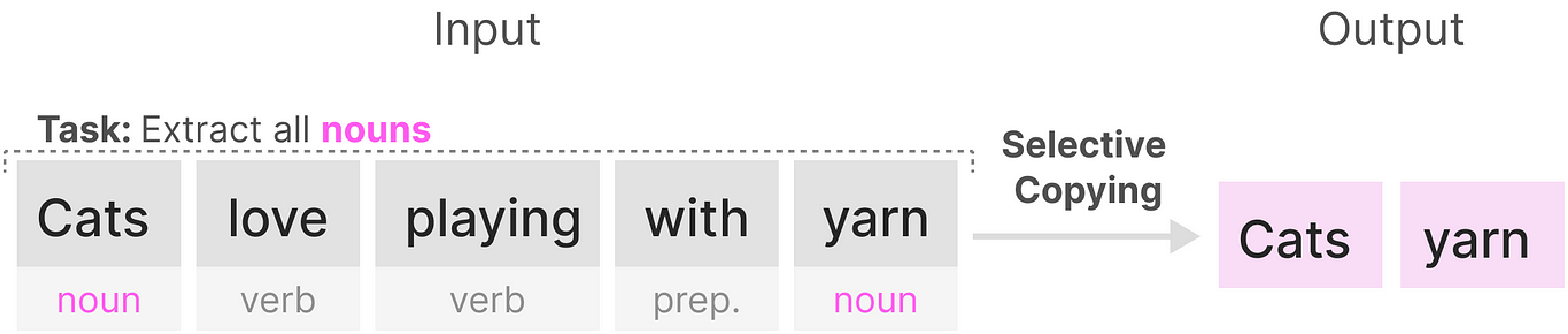

Hello
!
My
name
is
1
2
3
4
5
Input
Mamba Block
Repeated *n* times
RMS Norm
skip connection
projection
projection
Convolution
Silu
activation
Silu
Selective SSM
X
activation or multiplication
projection
+
RMS Norm
Linear + Softmax
Maarten
6
Output

Experiments

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

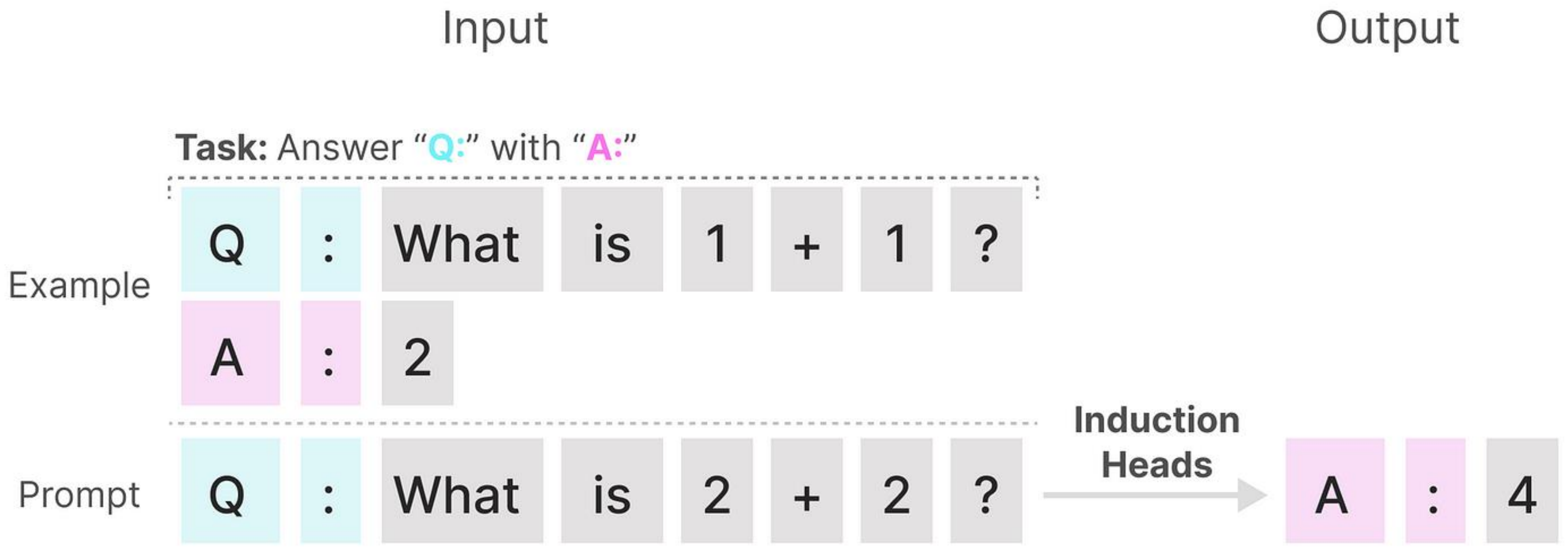
Experiments

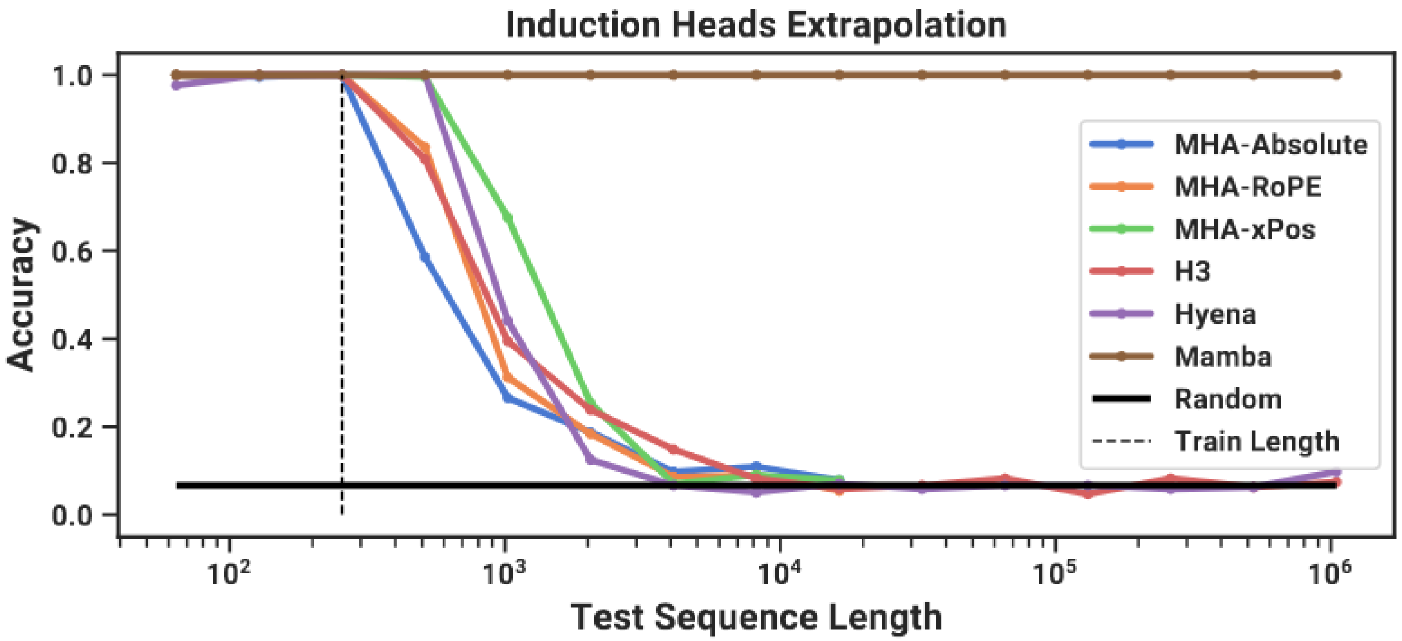


Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

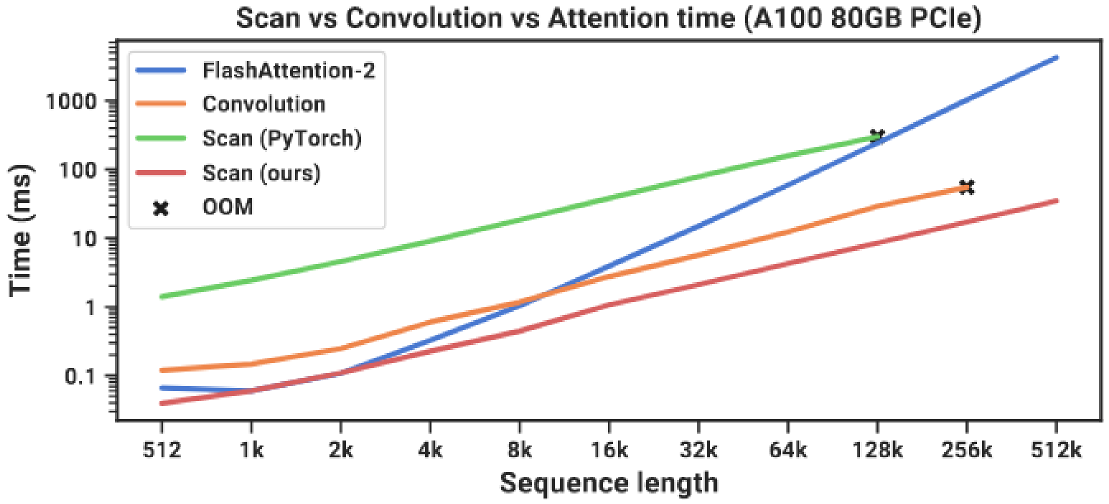
Table 1: (**Selective Copying**.) Accuracy for combinations of architectures and inner sequence layers.

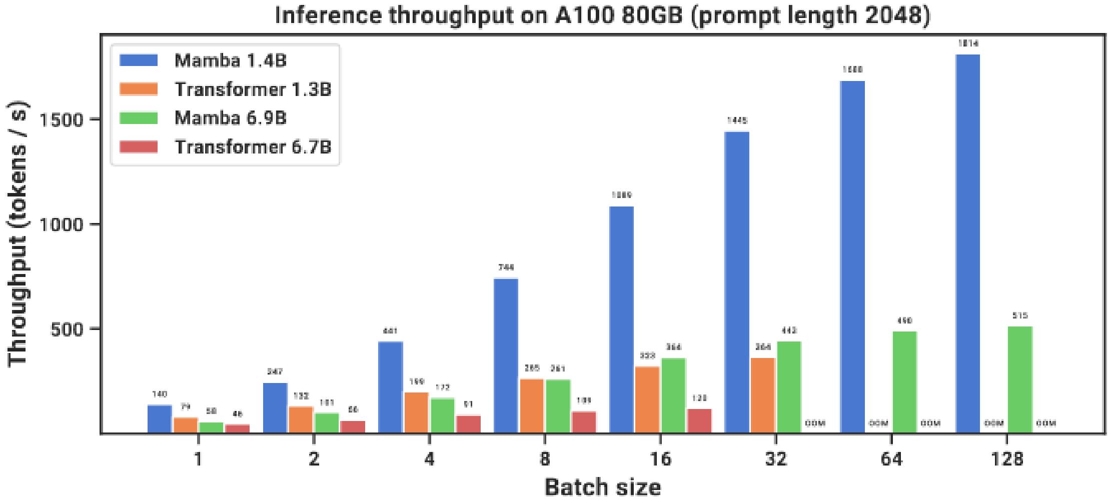
Experiments





Experiments





Experiments

Table 6: (**Ablations: Architecture and SSM layer**.) The Mamba block performs similarly to H3 while being simpler. In the inner layer, there is little difference among different parameterizations of LTI models, while selective SSMs (S6) provide a large improvement. More specifically, the S4 (real) variant is S4D-Real and the S4 (complex) variant is S4D-Lin.

Model	Arch.	SSM Layer	Perplexity
Hyena	H3	Hyena	10.24
H3	H3	S4 (complex)	10.30
-	H3	S4 (real)	10.34
-	H3	S6	**8.95**

Model	Arch.	SSM Layer	Perplexity
-	Mamba	Hyena	10.75
-	Mamba	S4 (complex)	10.54
-	Mamba	S4 (real)	10.56
Mamba	Mamba	S6	**8.69**

Why Rejected by ICLR24

- Need comparison with H3

Model	Lambada	HellaSwag	PIQA	Arc-E	Arc-C	WinoGrande	Avg
Hybrid H3-130M	25.8	31.7	64.2	44.4	24.2	50.6	40.2
Mamba-130M	44.2	35.2	64.5	48.0	24.2	52.3	44.7
Hybrid H3-360M	48.0	41.5	68.1	51.4	24.7	54.1	48.0
Mamba-370M	55.6	46.5	69.5	55.1	28.0	55.3	51.7
Hybrid H3-1.3B	49.6	52.6	71.3	59.2	28.1	56.9	53.0
Mamba-1.4B	65.0	59.1	74.2	65.5	32.8	61.5	59.7
Hybrid H3-2.7B	55.7	59.7	73.3	65.6	32.3	61.4	58.0
Mamba-2.8B	69.2	66.1	75.2	69.7	36.3	63.5	63.3

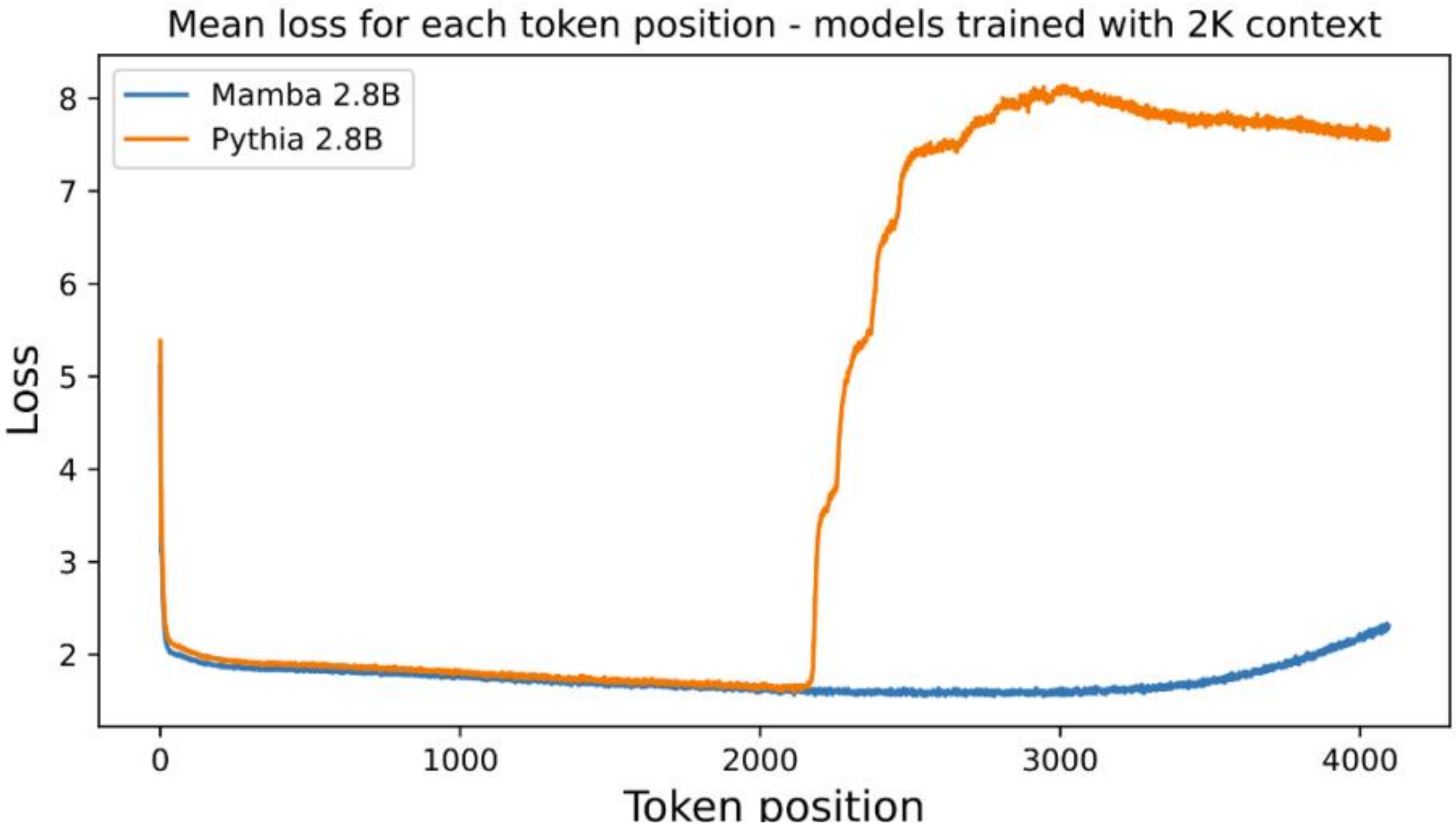
Why Rejected by ICLR24

- Scaling beyond 1.4B, vs Transformer 10B

Model	Lambada	HellaSwag	PIQA	Arc-E	Arc-C	WinoGrande	Avg
GPT-Neo 2.7B	62.2	55.8	72.1	61.1	30.2	57.6	56.5
Hybrid H3-2.7B	55.7	59.7	73.3	65.6	32.3	61.4	58.0
OPT-2.7B	63.6	60.6	74.8	60.8	31.3	61.0	58.7
Pythia-2.8B	64.7	59.3	74.0	64.1	32.9	59.7	59.1
RWKV-3B	63.9	59.6	73.7	67.8	33.1	59.6	59.6
Mamba-2.8B	**69.2**	*66.1*	*75.2*	**69.7**	*36.3*	*63.5*	**63.3**
OPT-6.7B	*67.7*	**67.2**	**76.3**	65.6	34.9	**65.5**	*62.9*
Pythia-6.9B	67.1	64.0	*75.2*	67.3	35.5	61.3	61.7
RWKV-7.4B	67.2	65.5	76.1	*67.8*	**37.5**	61.0	62.5

Why Rejected by ICLR24

- Beyond Training Length



Conclusions

- Selection Mechanism
- Hardware-aware Algorithm
- Simpler SSM Architecture

Thank You！