Mamba: Linear-Time Sequence Modeling with Selective State Spaces

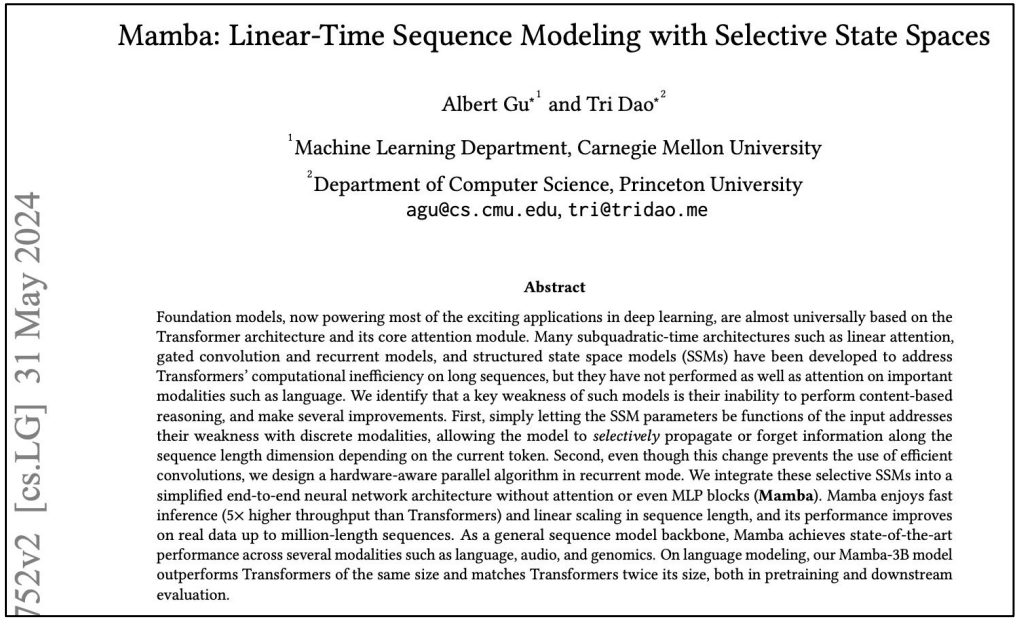
연세대학교 통계데이터사이언스학과

통합과정 8학기 이승한

Contents

1. Limitation of Transformer
2. State Space Models (SSM)
3. Linear SSM
4. MAMBA - Selective SSM

(2024.06.24 기준) 437회 인용



Mamba: Linear-Time Sequence Modeling with Selective State Spaces

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

[1]Machine Learning Department, Carnegie Mellon University

[2]Department of Computer Science, Princeton University

agu@cs.cmu.edu, tri@tridao.me

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to *selectively* propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (**Mamba**). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

참고 자료

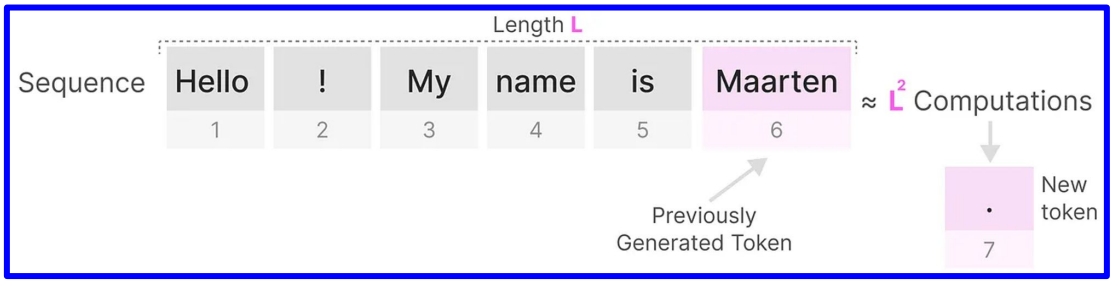
- A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," arXiv preprint arXiv:2312.00752, 2023.
- https://youtu.be/JjxBNBzDbNk
- https://tulip-phalange-a1e.notion.site/05f977226a0e44c6b35ed9bfe0076839

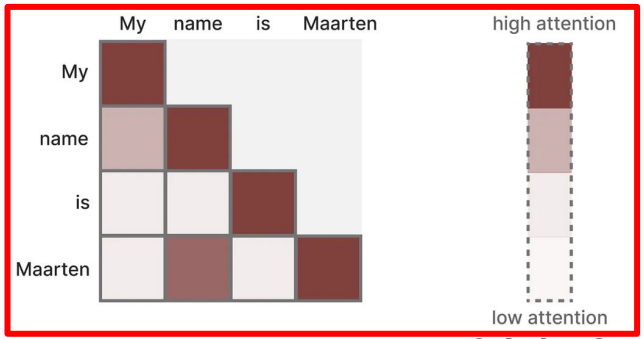
1. Limitation of Transformer

1. Limitation of Transformer

(1) Transformer의 특징

- 특징 1) 전체 Sequence에 대해 하나의 context vector로 압축할 필요가 없음.
  들어오는 Input이 필요한 것을 알아서 취사선택하면 되므로!
- 특징 2) 빠른 훈련 속도 (병렬화 가능)
- 특징 3) 느린 추론 속도 (O(L^2)의 complexity)



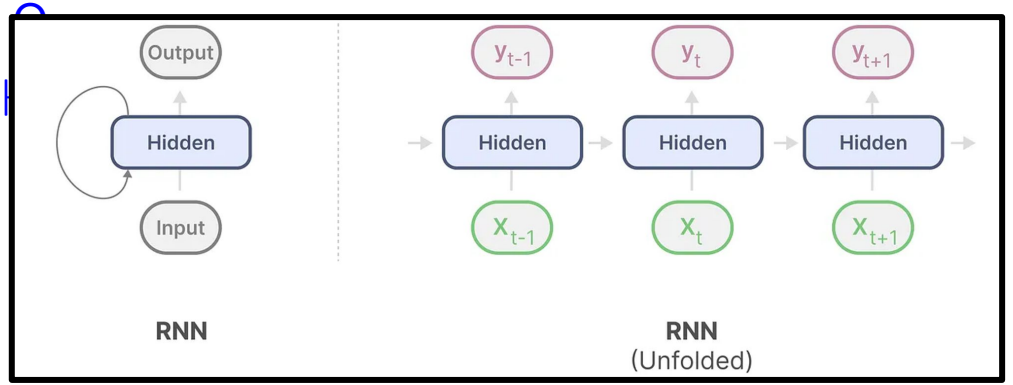


1. Limitation of Transformer

(2) RNN의 특징

- 특징 1) Recursive strategy
  - 과거의 hidden state + 현재의 input을 입력으로 받음
- 특징 2) 전체 Sequence에 대해 하나의 context vector로 압축.
  - 오래 전 과거 정보 손실 가능성 O
- 특징 3) 느린 훈련 속도 (병렬화 불가
- 특징 4) 빠른 추론 속도

  (O(L)의 complexity)



1. Limitation of Transformer

(3) Transformer vs. RNN

	Training 속도	Inference 속도
Transformer	**Fast**	Slow
RNN	Slow	**Fast**
MAMBA	**Fast**	**Fast**

2. State Space Models (SSM)

2. State Space Model (SSM)

(1) SSM의 기본 특징

- Sequence 데이터를 다루는 모델 (i.e. Text, Time Series)
- 상태를 설명하는 모델
  - 상태 = (현재 상태 -> 행동 -> 미래 상태)
- 즉, "다음 상태가 어떻게 될지" 예측하는 모델

x(t): Input sequence

h(t): Hidden state

y(t): Output sequence

2. State Space Model (SSM)

(2) 상태 방정식

- 상태가 어떻게 변화할지를 나타내는 방정식

State equation $$\mathbf{h'}(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t)$$

Output equation $$\mathbf{y}(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{x}(t)$$

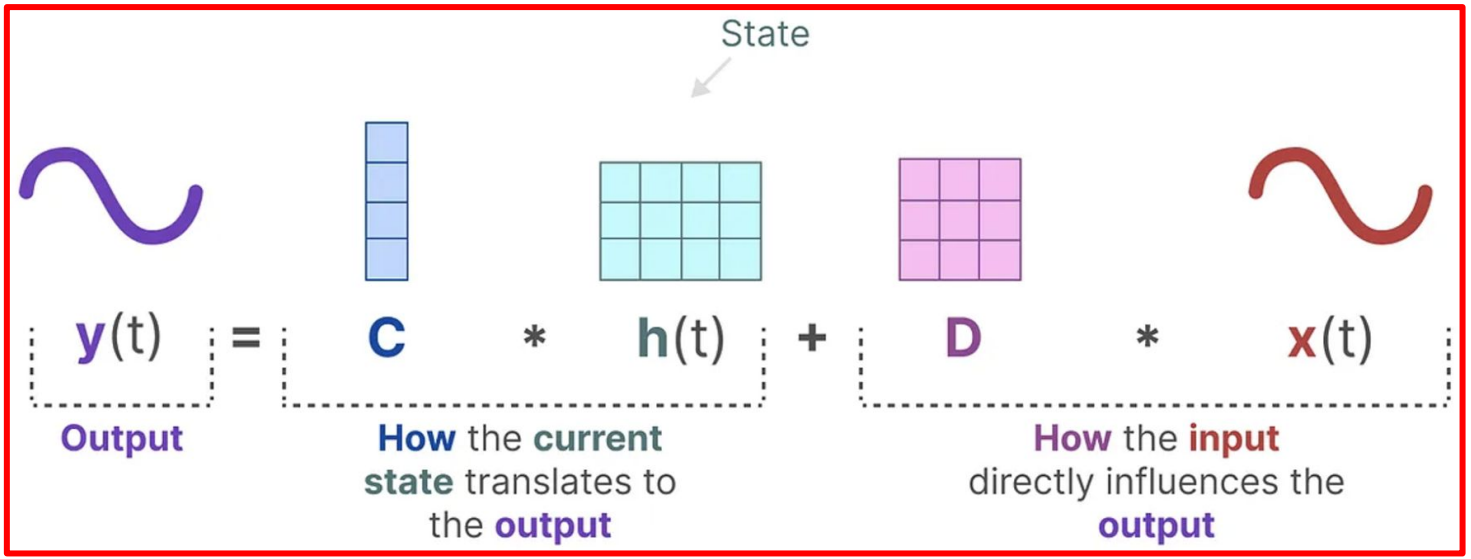
=> Input sequence와 Hidden state를 통해 Output sequence를 예측하는 원칙을 밝힐 수 있음

2. State Space Model (SSM)

(2) 상태 방정식

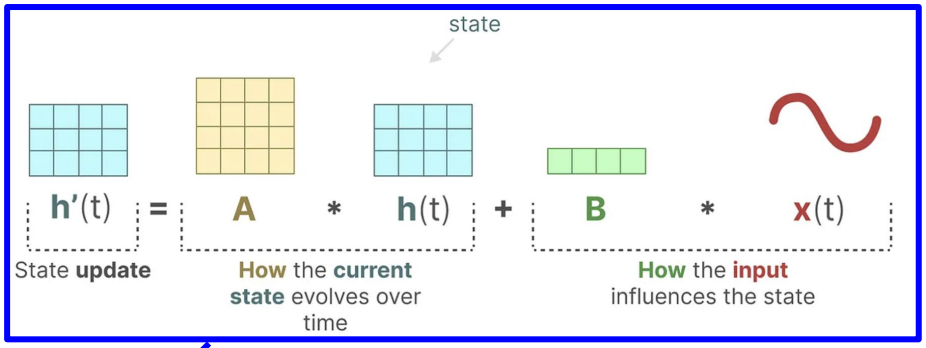
State equation	$\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$
Output equation	$\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

Output equation



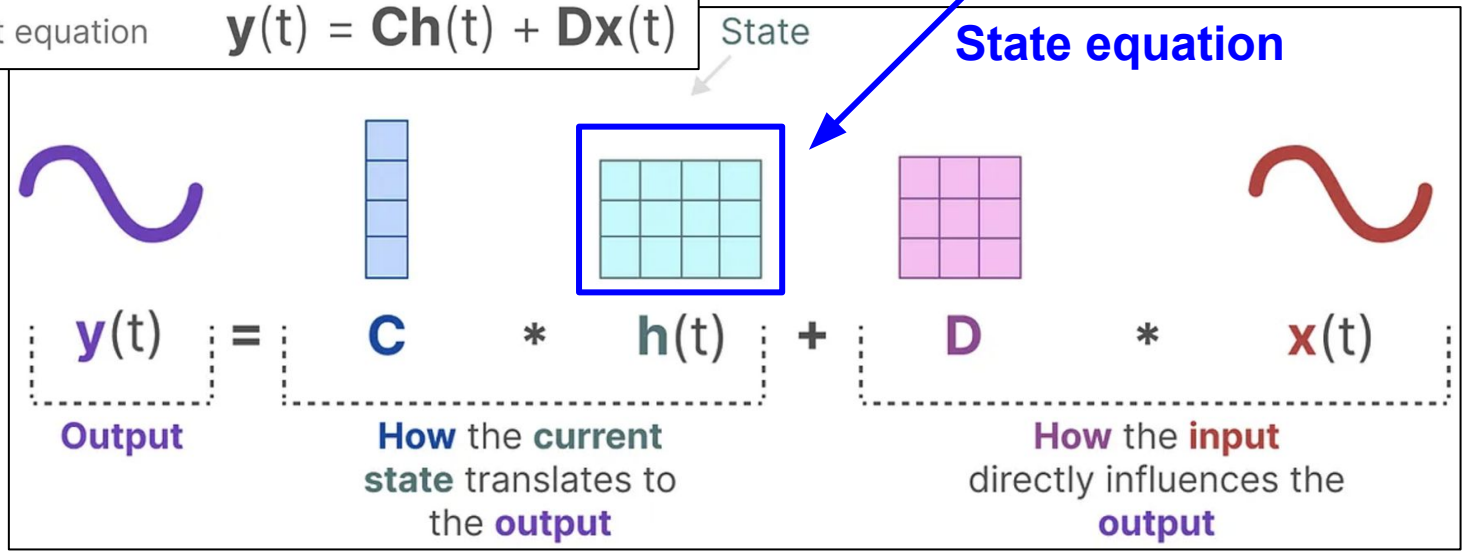
2. State Space Model (SSM)

(2) 상태 방정식



State equation $h'(t) = Ah(t) + Bx(t)$

Output equation $y(t) = Ch(t) + Dx(t)$

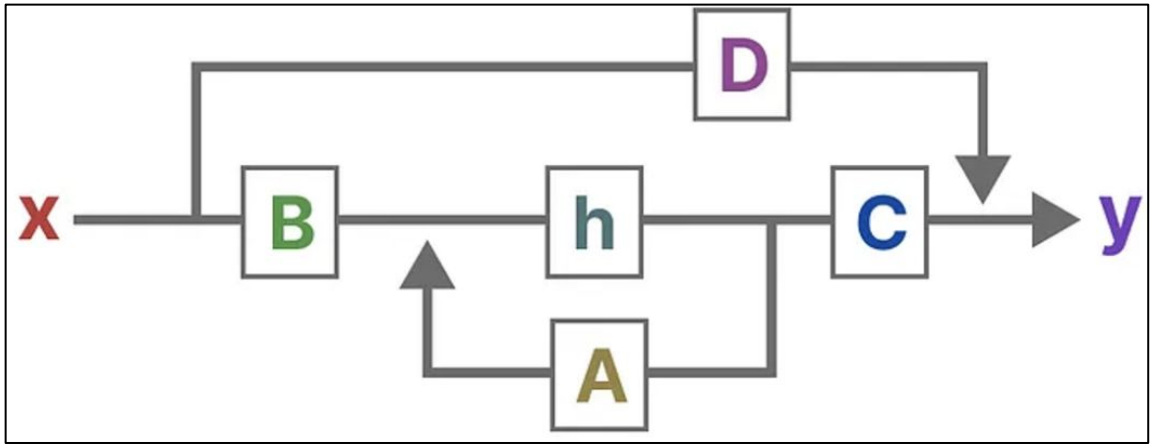


2. State Space Model (SSM)

(2) 상태 방정식

State equation $\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$

Output equation $\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$



2. State Space Model (SSM)

State equation $h'(t) = Ah(t) + Bx(t)$

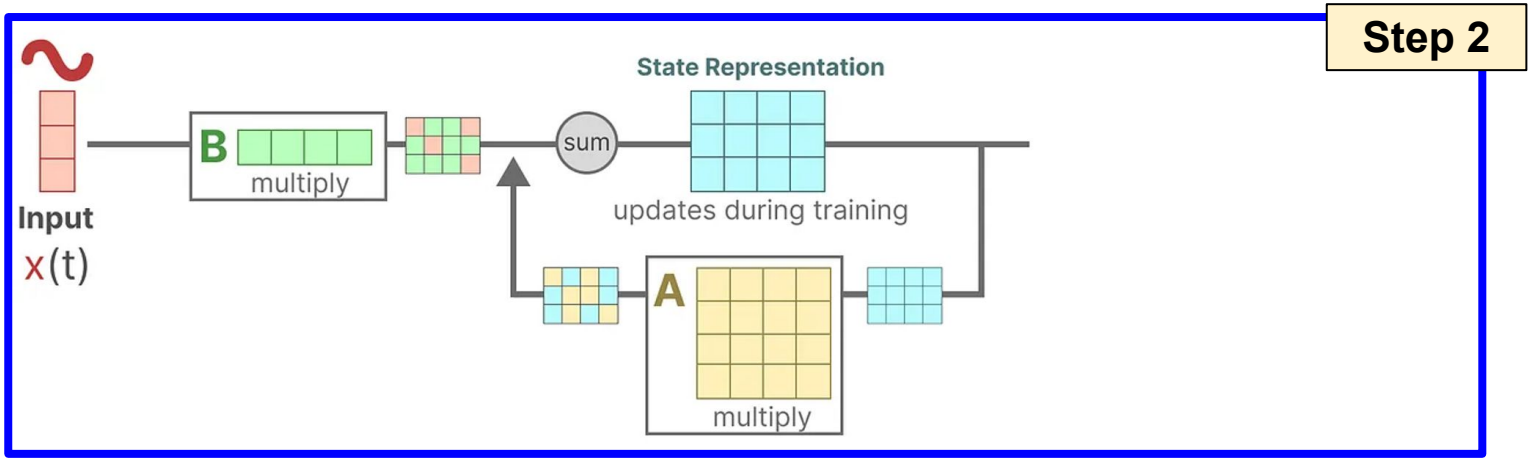
Output equation $y(t) = Ch(t) + Dx(t)$

(3) 학습 과정

2. State Space Model (SSM)

State equation	$\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$
Output equation	$\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

(3) 학습 과정

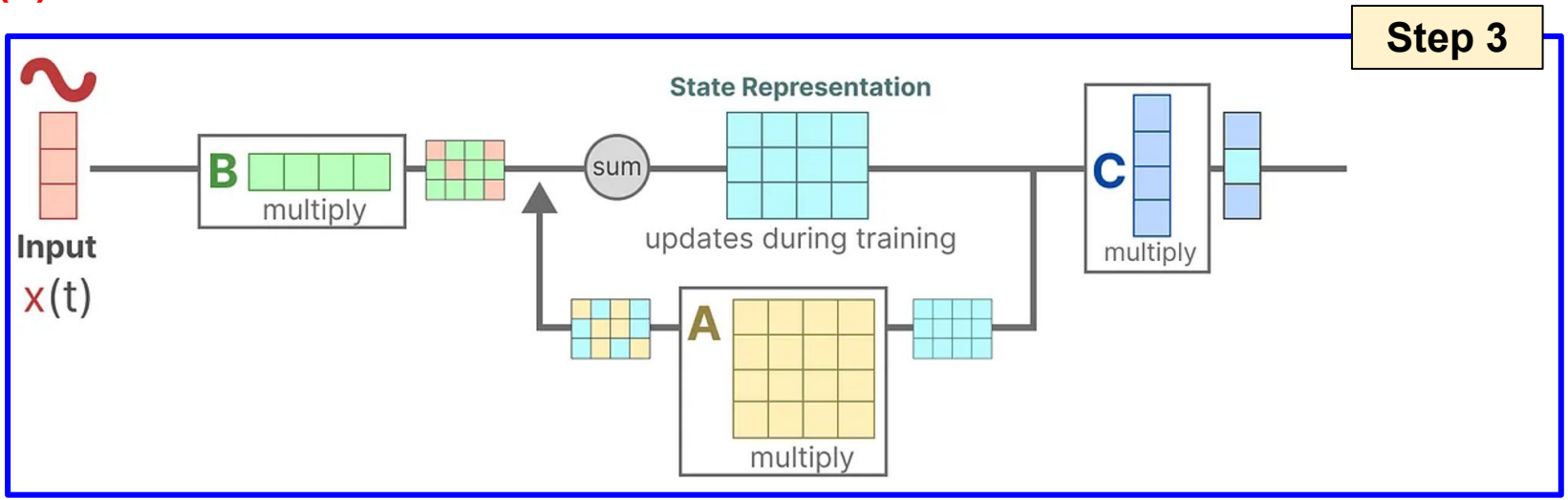


2. State Space Model (SSM)

State equation $h'(t) = Ah(t) + Bx(t)$

Output equation $y(t) = Ch(t) + Dx(t)$

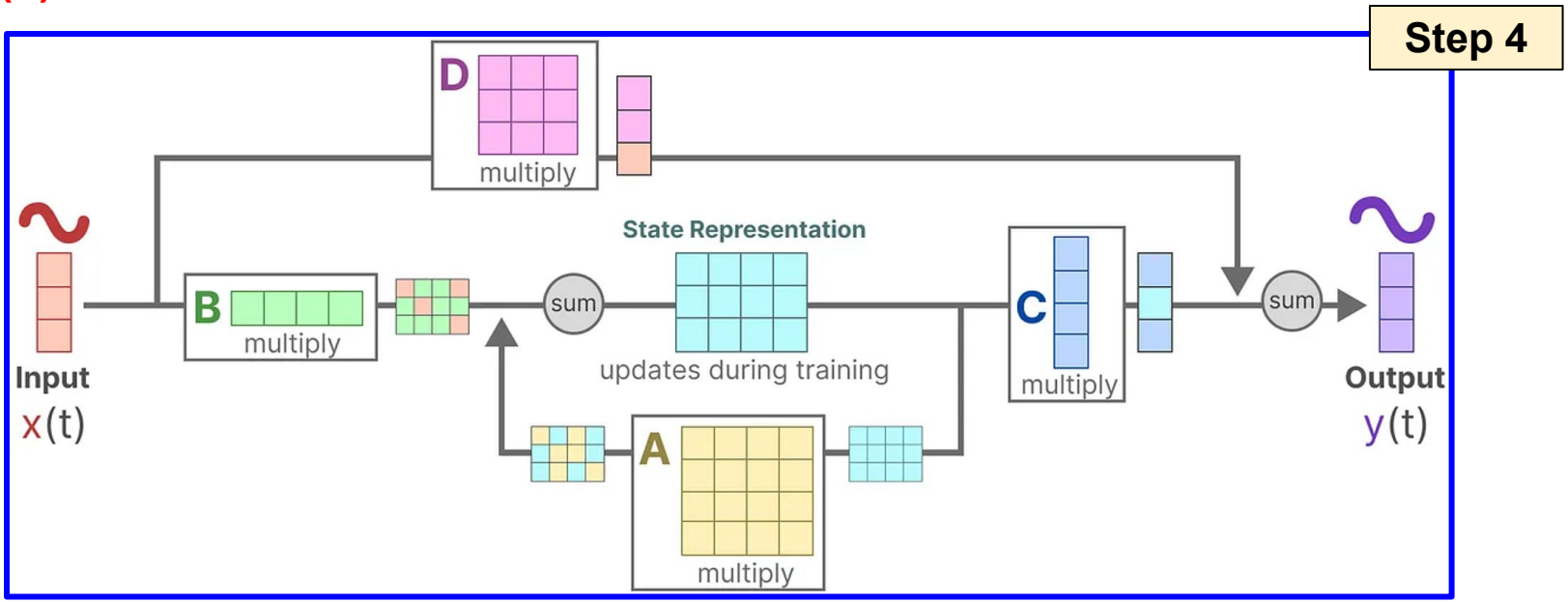
(3) 학습 과정



2. State Space Model (SSM)

State equation	$\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$
Output equation	$\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

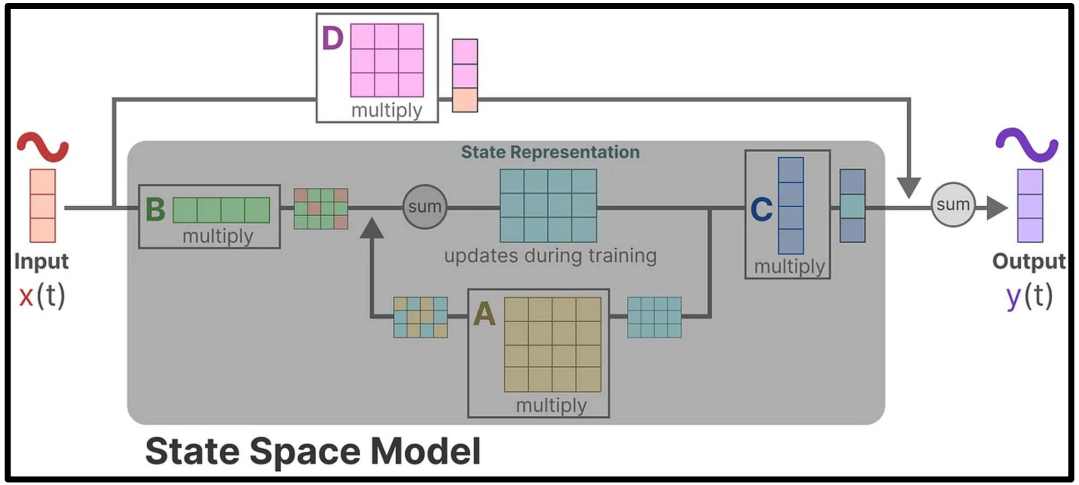
(3) 학습 과정

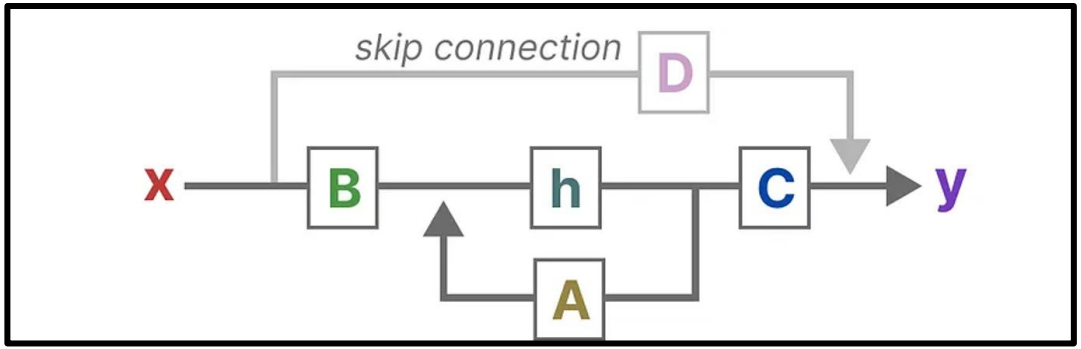


2. State Space Model (SSM)

State equation	$\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$
Output equation	$\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

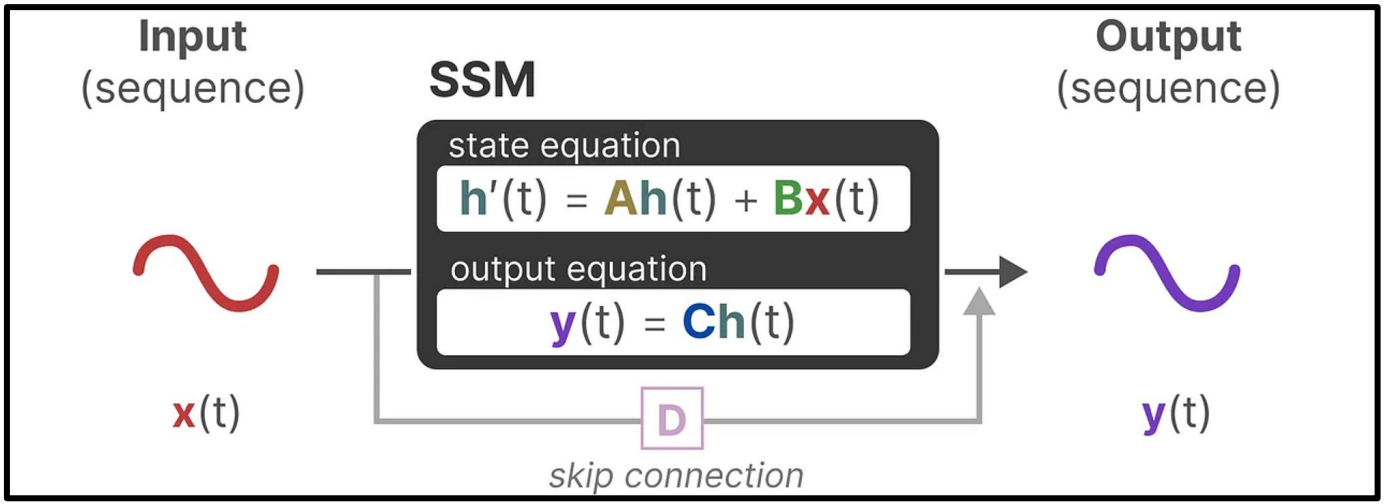
(4) SSM 요약





2. State Space Model (SSM)

(4) SSM 요약



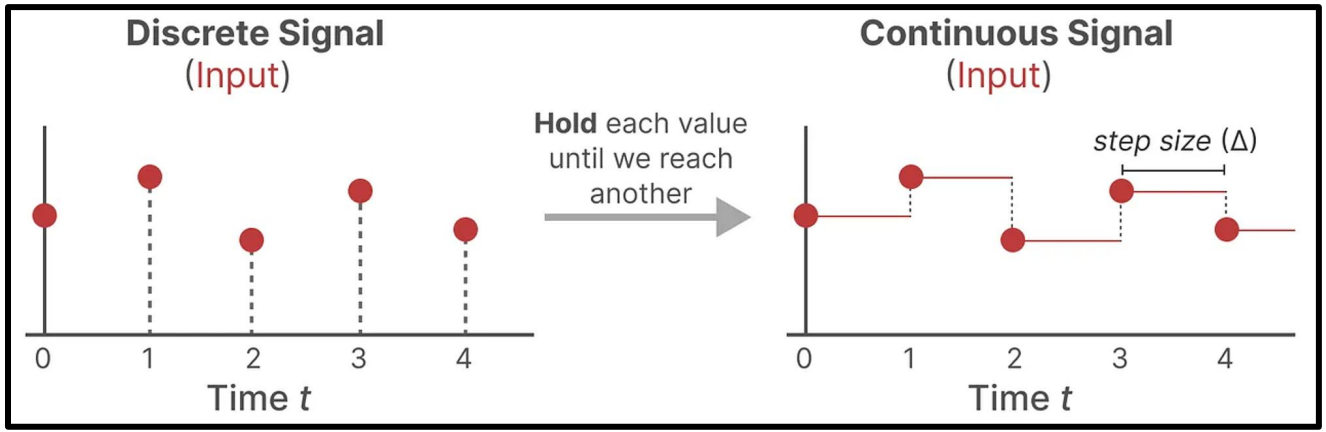
- 관측된 데이터 (X(t), h(t))로부터 시스템의 상태를 예측
- Continuous-time representation

2. State Space Model (SSM)

(5) Continuous -> Discrete

제로-오더 홀드(Zero-order hold)

- 새로운 이산 신호를 받을 때까지 그 값을 유지
- 값을 유지하는 시간 길이 = (학습 가능한 매개변수인) 스텝 크기 Δ



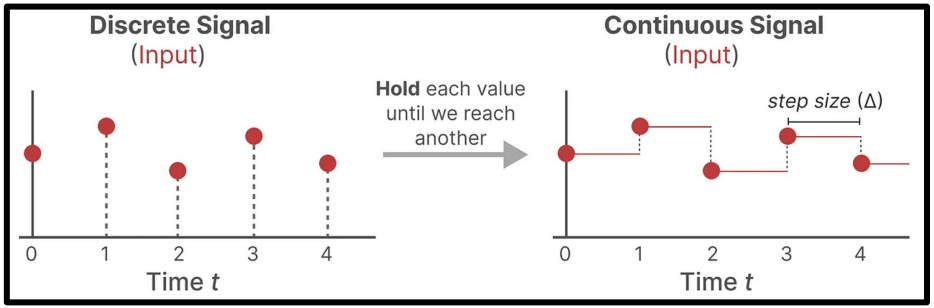
2. State Space Model (SSM)

(5) Continuous -> Discrete

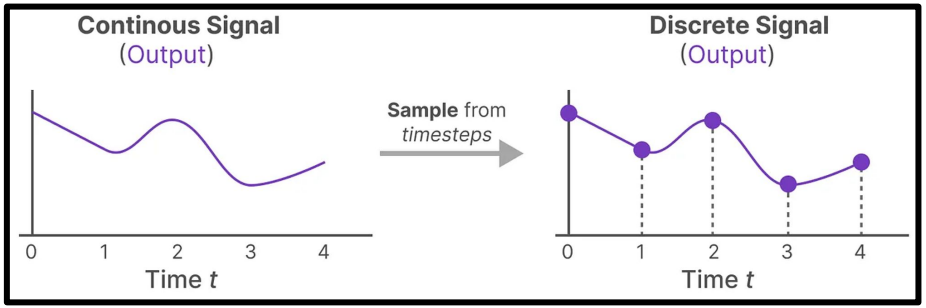
Input & Output

- Input: Discrete -> Continuous
- Output: continuous -> Discrete

최종 Output

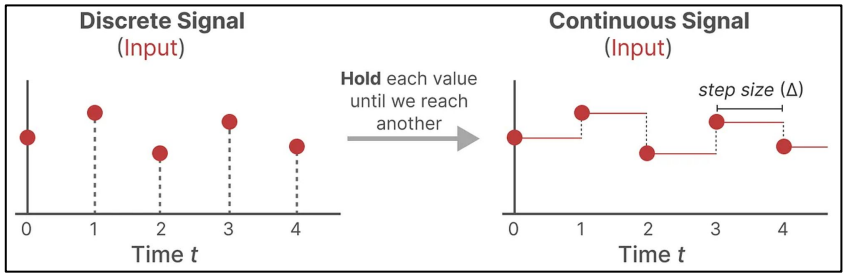


Input



Output

2. State Space Model (SSM)



(5) Continuous -> Discrete

제로-오더 홀드(Zero-order hold)

- 수학적 표현

Discretized matrix **A**

$$\overline{\mathbf{A}} = \exp(\Delta \mathbf{A})$$

Discretized matrix **B**

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

2. State Space Model (SSM)

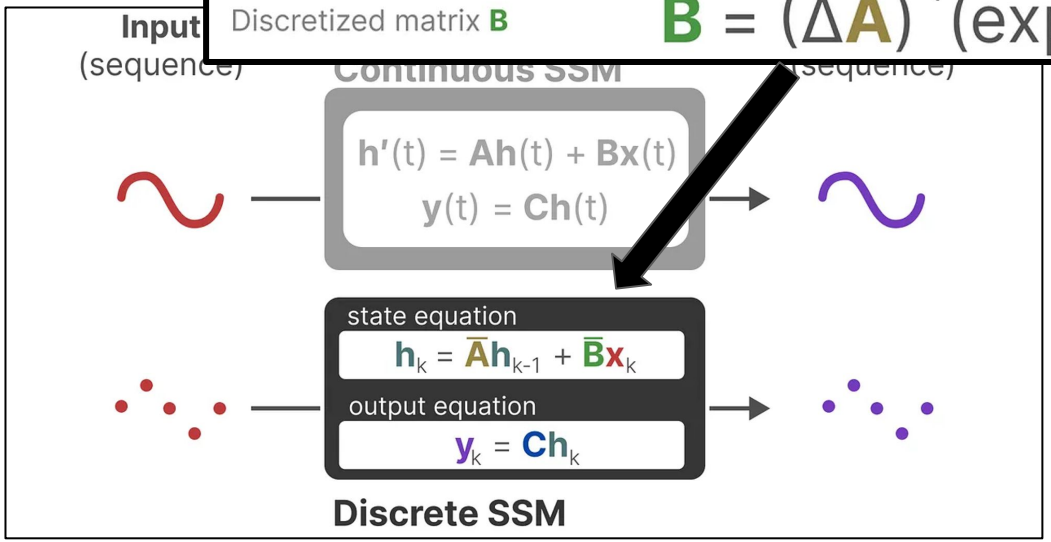
(5) Continuous -> Discre

Discretized matrix **A**

$$\overline{\mathbf{A}} = \exp(\Delta \mathbf{A})$$

Discretized matrix **B**

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

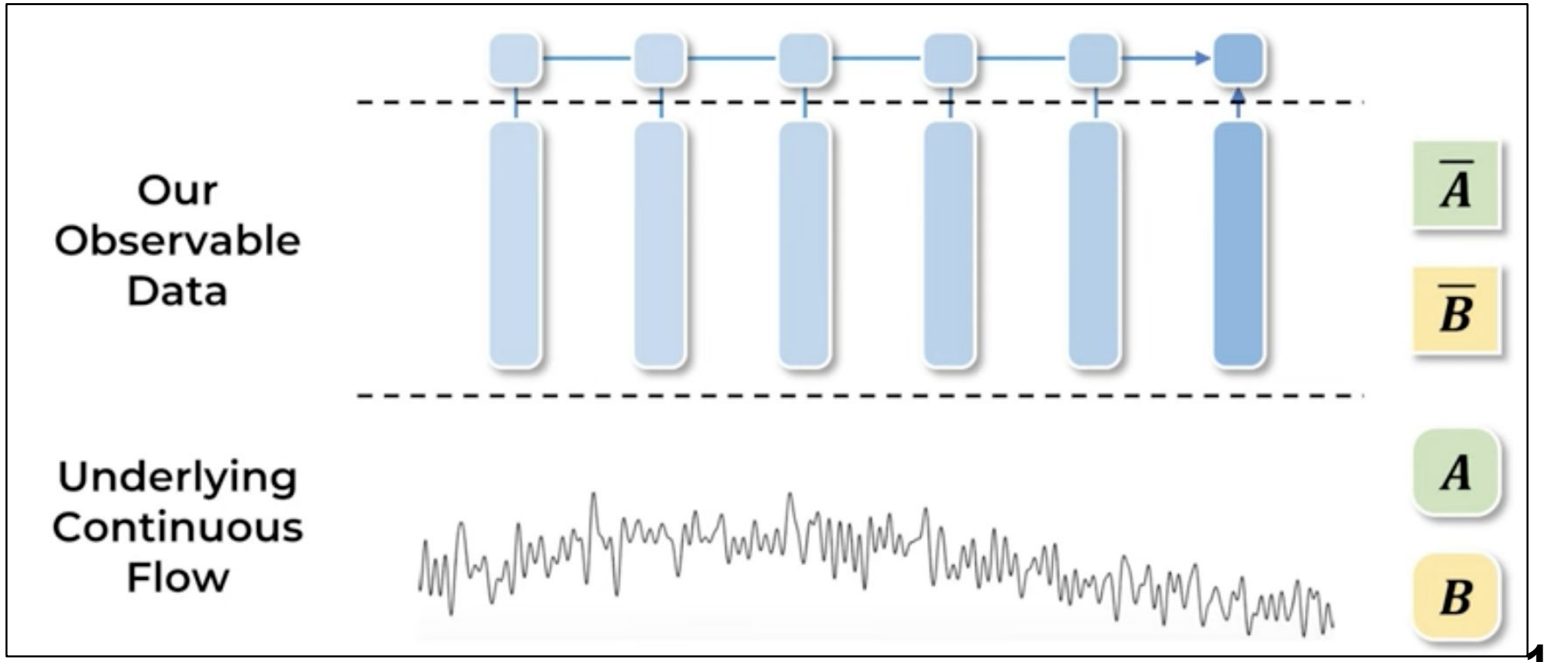


- A,B: Model parameters
- t (k) : Continuous (Discrete) step

2. State Space Model (SSM)

(5) Continuous -> Discrete

Parameter: A , B (not $\bar{A}$, $\bar{B}$)



2. State Space Model (SSM)

state equation

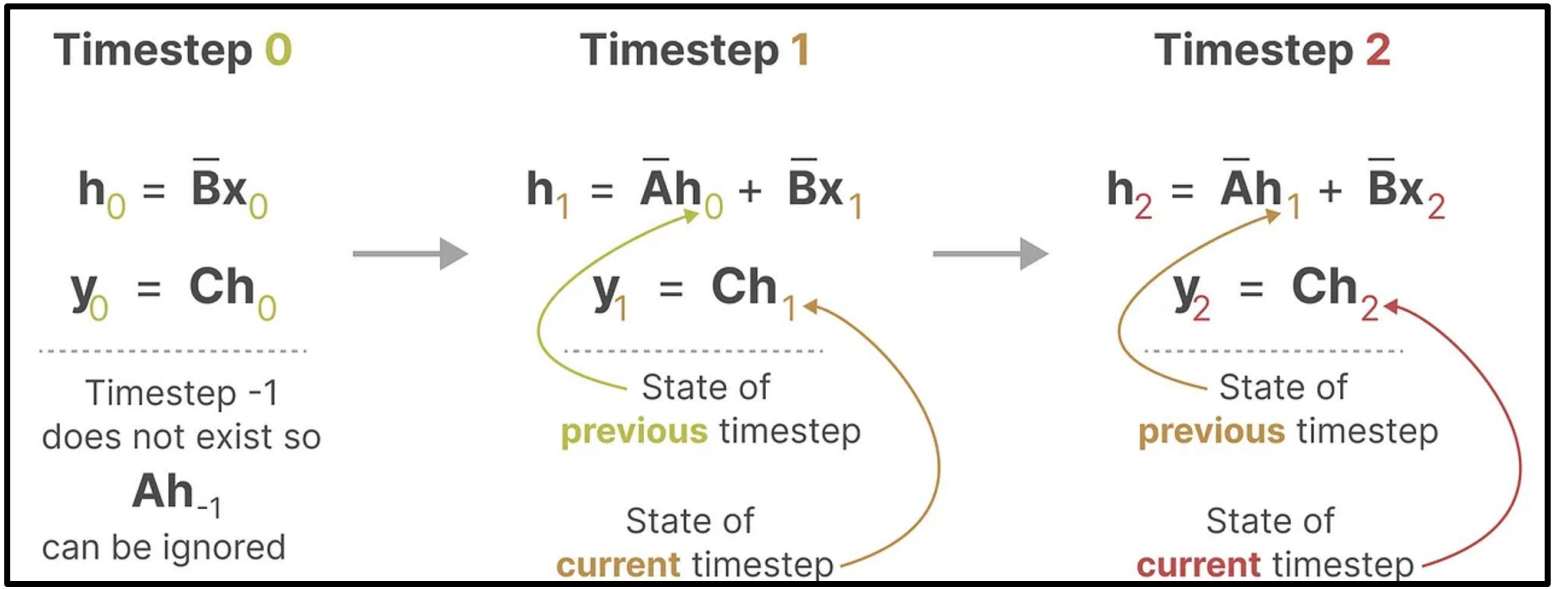
$$h_k = \bar{A}h_{k-1} + \bar{B}x_k$$

output equation

$$y_k = Ch_k$$

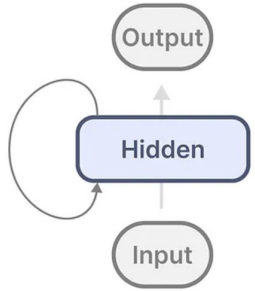
(6) Recursive Expression

for k in [0,K]:

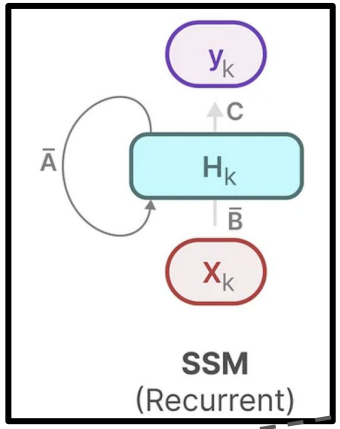


2. State Space Model (SSM)

(6) Recursive Expression

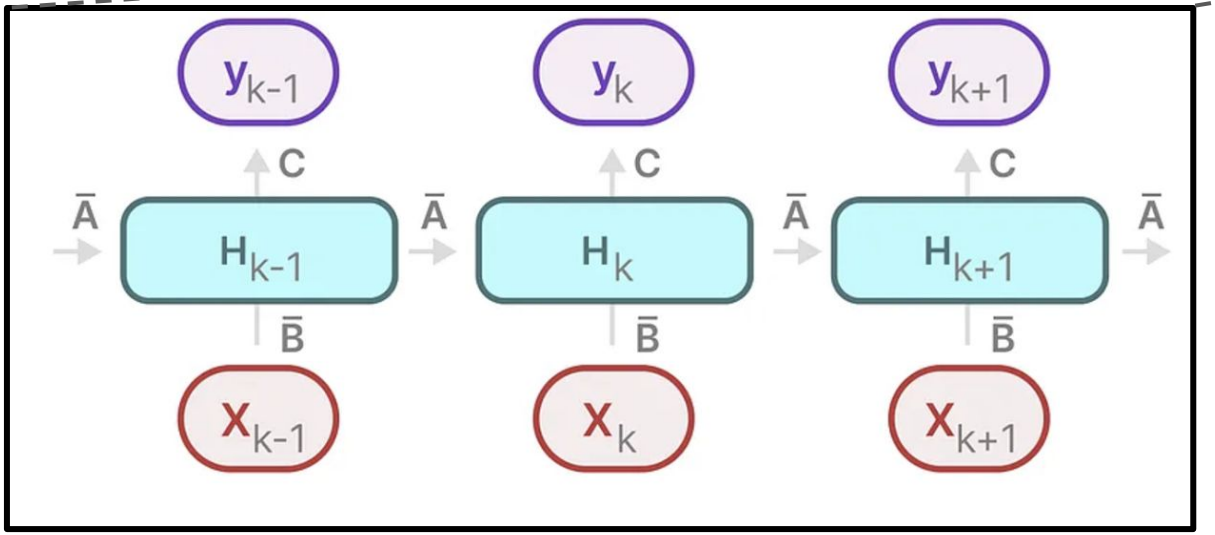


RNN



SSM
(Recurrent)

for k in [0,K]:



Fast Inference

Slow Training

(like RNN)

2. State Space Model (SSM)

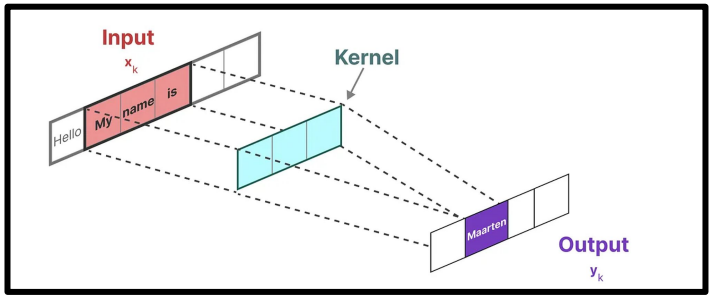
State equation $\mathbf{h'}(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$

Output equation $\mathbf{y}(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

(7) Convolutional Expression

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

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



1D convolution

$$\text{kernel} \longrightarrow \overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}^{k}\mathbf{B}}, \ldots)$$

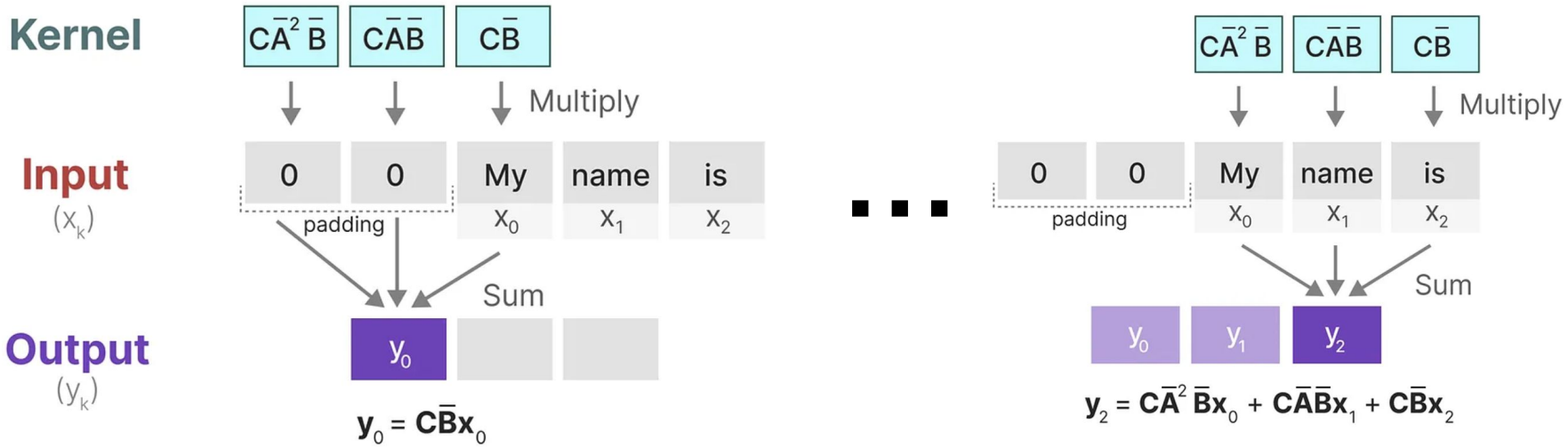
$$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}$$

output input kernel

Convolution kernel of SSM

2. State Space Model (SSM)

(7) Convolutional Expression

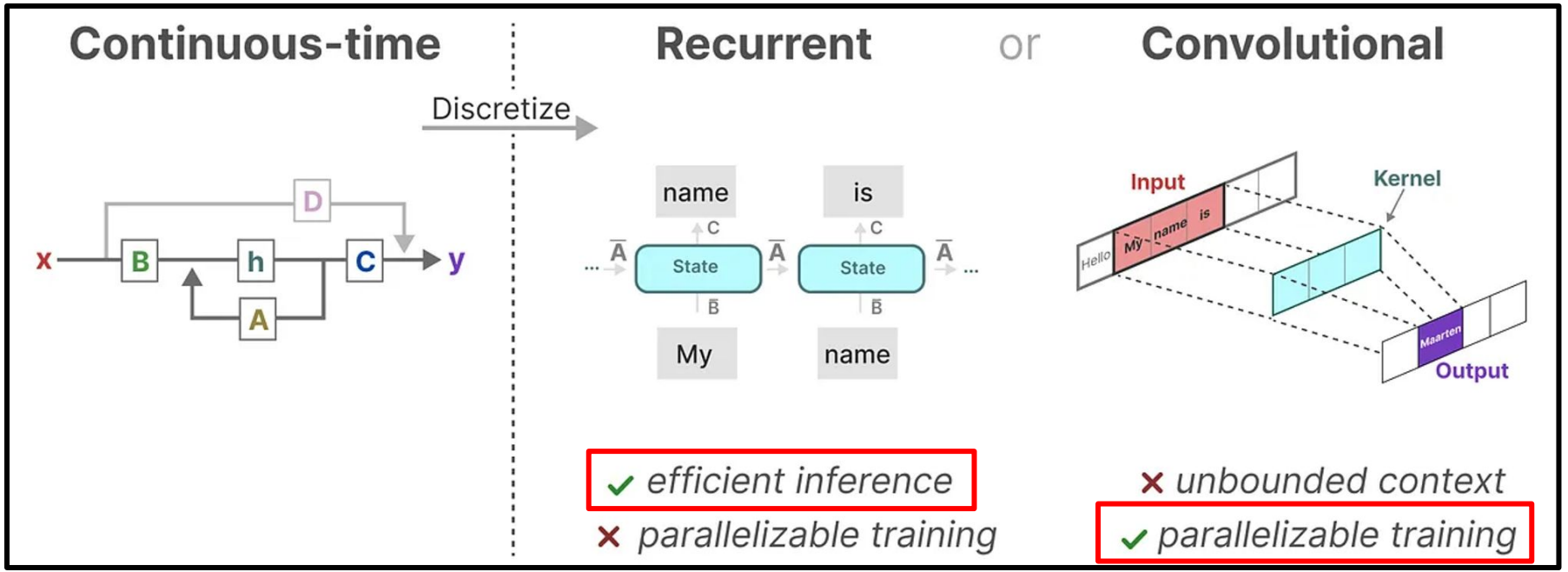


Convolution으로 표현함으로써 병렬적으로 훈련 가능

But 유한한 추론 (고정된 kernel size)

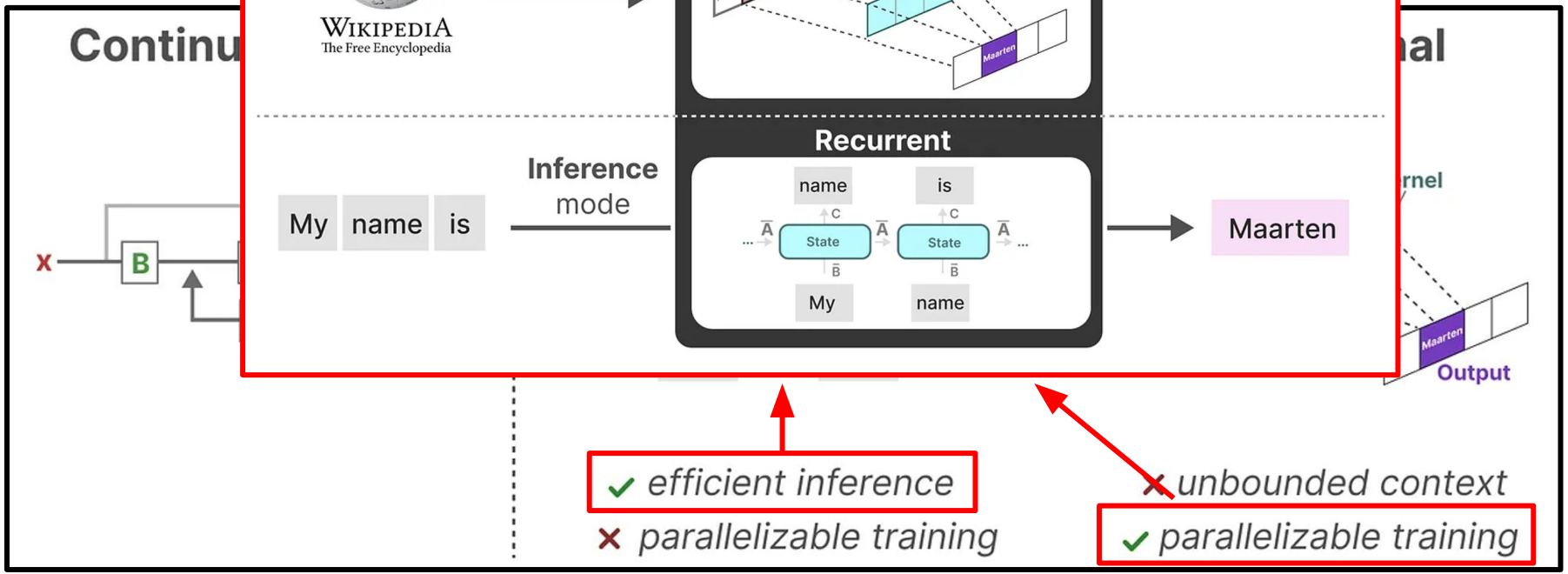
2. State Space Model (SSM)

(8) 세 가지 표현 방식



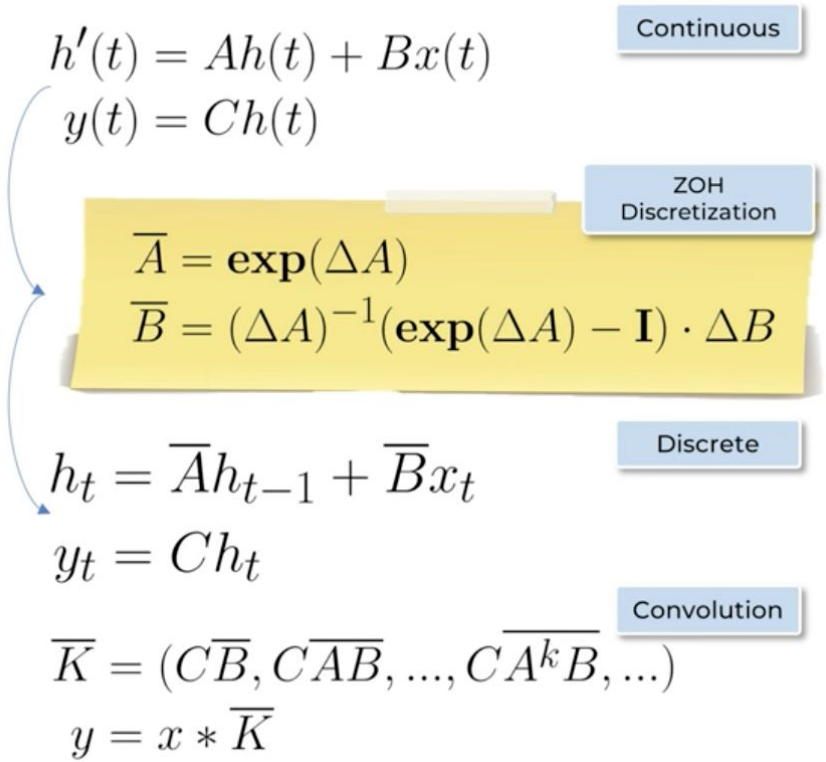
2. State S

(8) 세 가지 표



2. State Space Model (SSM)

(9) Summary



3. Linear SSM (LSSM)

3. Linear SSM (LSSM)

(1) LSSM의 특징

- 모델: Stack SSM blocks! (w/o MLP)
- 특징
  - (1) Fast Training (as CNN)
  - (2) Fast Inference (as RNN)
- 핵심: A matrix

State equation $\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t)$

Output equation $\mathbf{y}(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{x}(t)$

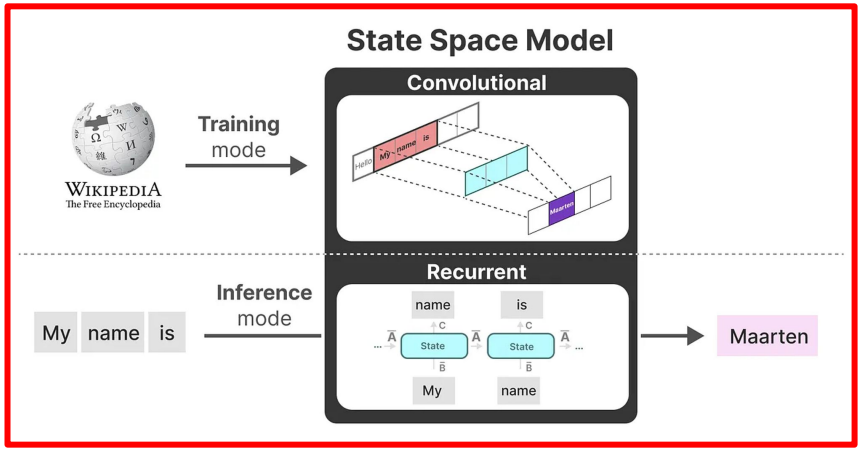
3. Linear SSM (LSSM)

(2) 선형 시간 불변성 (Linear Time Invariance, LTI)

- SSM의 parameter (A,B,C)가 t에 independent

  (= 생성하는 모든 토큰에 대해 동일한 파라미터)

- 정적 표현 (Static representation)
  - 내용 인식(content-awareness)이 없음



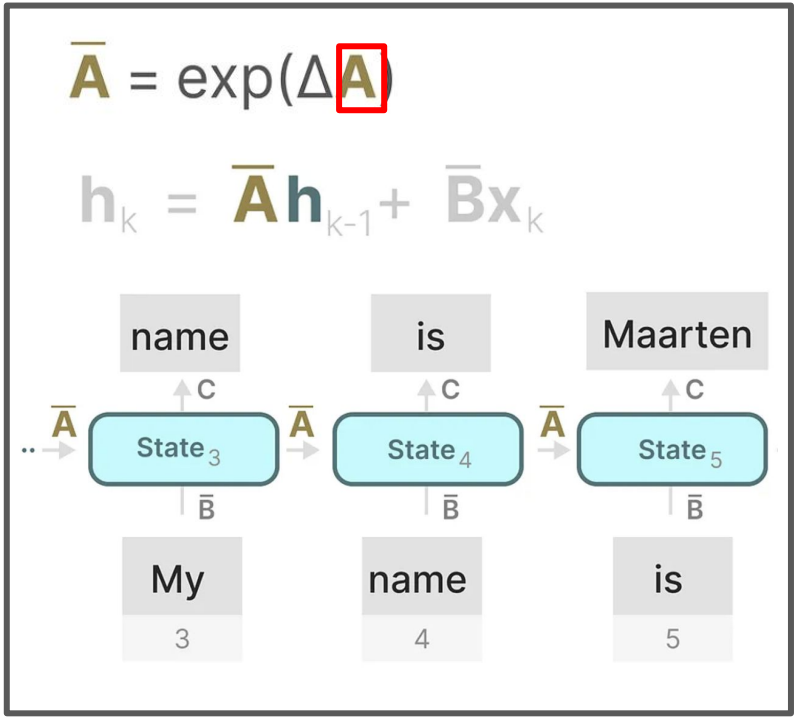
3. Linear SSM (LSSM)

(3) 행렬 A

State equation $h'(t) = Ah(t) + Bx(t)$

Output equation $y(t) = Ch(t) + Dx(t)$

이전 상태를 활용하여 새로운 상태 구축



3. Linear SSM (LSSM)

(3) 행렬 A

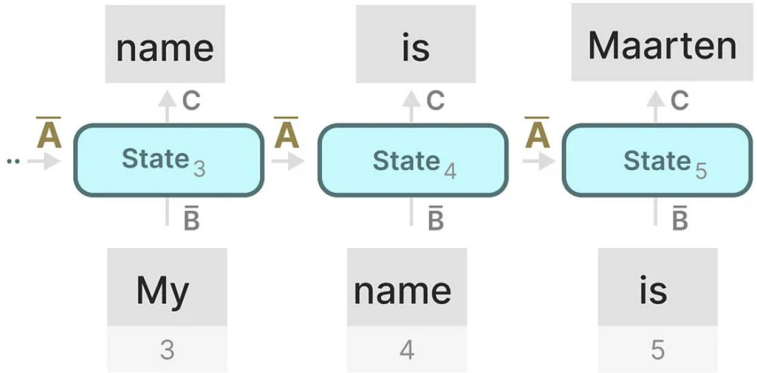
State equation $\quad h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$

Output equation $\quad y(t) = \mathbf{C}h(t) + \mathbf{D}x(t)$

이전 상태를 활용하여 새로운 상태 구축

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

$$h_k = \bar{\mathbf{A}} h_{k-1} + \bar{\mathbf{B}} x_k$$



Question) 어떻게 큰 메모리(맥락 크기)를 유지하는 방식으로 **A**를 생성?

3. Linear SSM (LSSM)

(4) HiPPO (High-order Polynomial Projection Operators)

"A matrix를 잘 초기화(initialization) 하자!"

State equation $\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t)$

Output equation $\mathbf{y}(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{x}(t)$

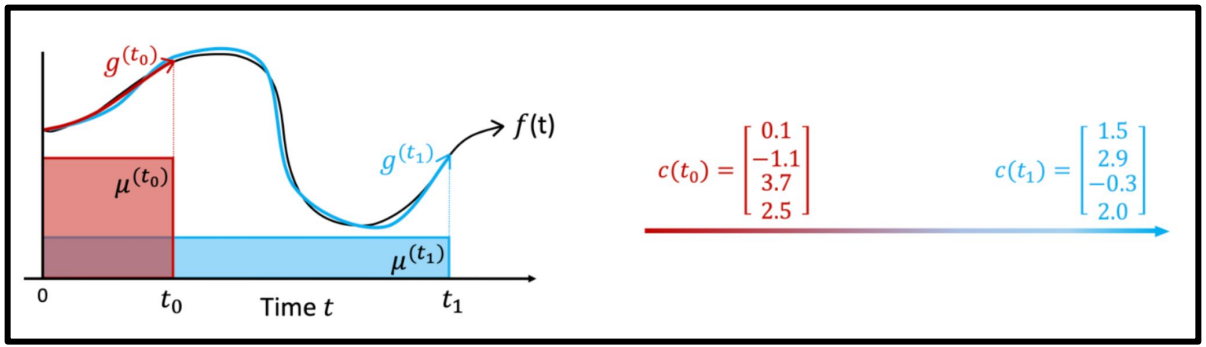
3. Linear SSM (LSSM)

(4) HiPPO (High-order Polynomial Projection Operators)

"지금까지 본 모든 입력 신호를 계수의 벡터로 압축"

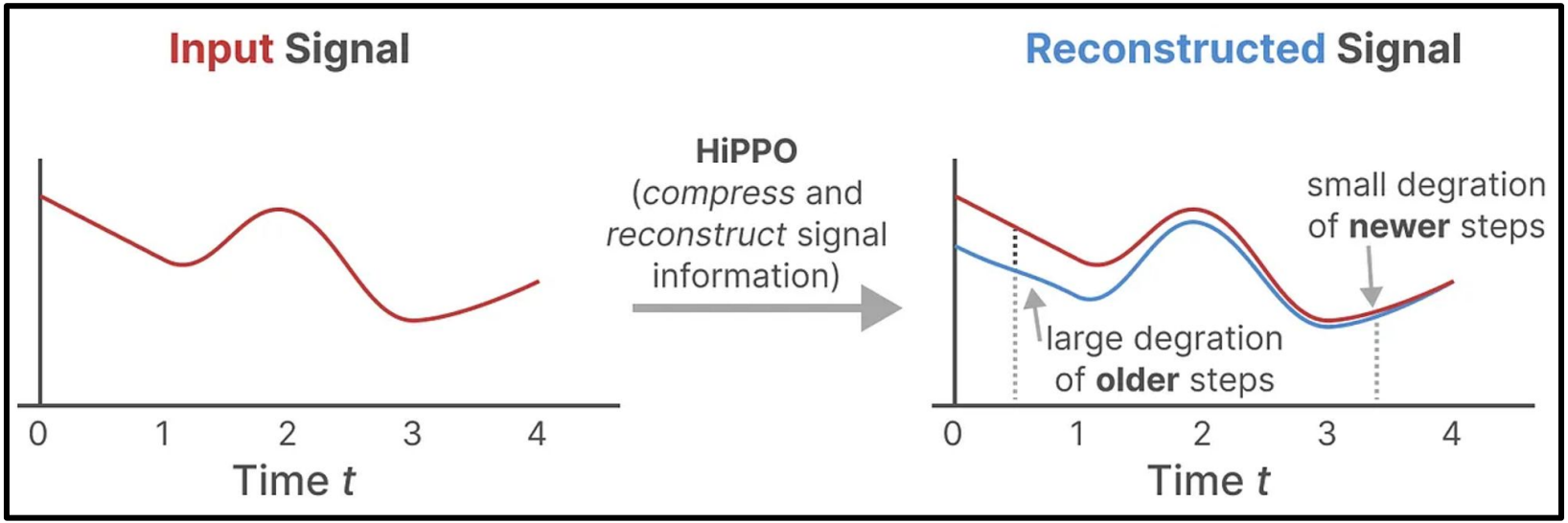
Online function approximation

- Ground truth: f(t)
- Approximation: g(t)
  - orthogonal polynomial (OP) basis에 f(t)를 projection & reconstruction



3. Linear SSM (LSSM)

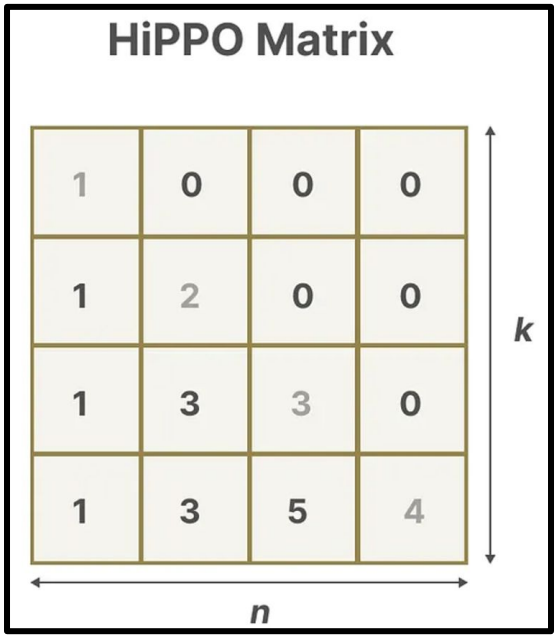
(4) HiPPO (High-order Polynomial Projection Operators)



HiPPO: 최근 토큰을 잘 포착 & 오래된 토큰을 감소시키는 상태 표현을 구축

3. Linear SSM (LSSM)

(4) HiPPO (High-order Polynomial Projection Operators)



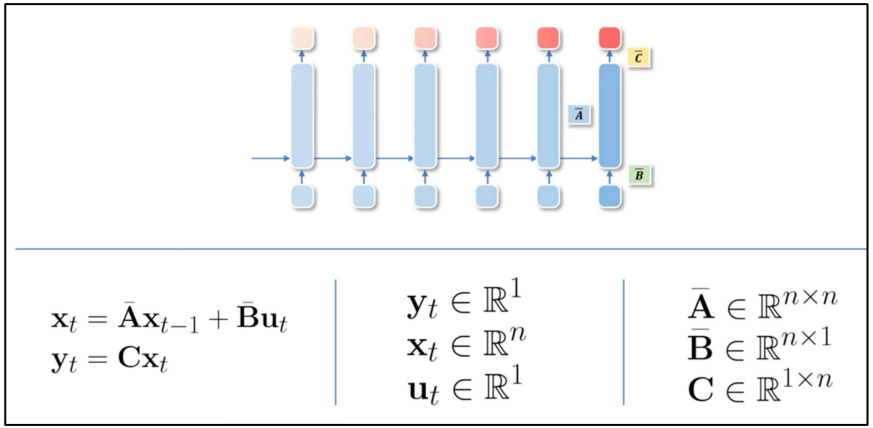
$$A_{nk} \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{everything below the diagonal} \\ n+1 & \text{the diagonal} \\ 0 & \text{everything above the diagonal} \end{cases}$$

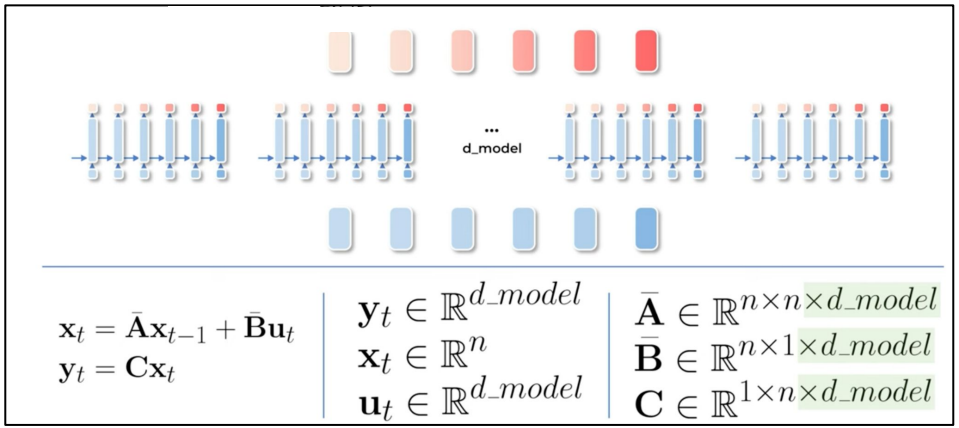
A w/ **HiPPO** > A w/ **random initialization**

3. Linear SSM (LSSM)

(5) LSSL의 Dimension 분석

*** Notation 유의**





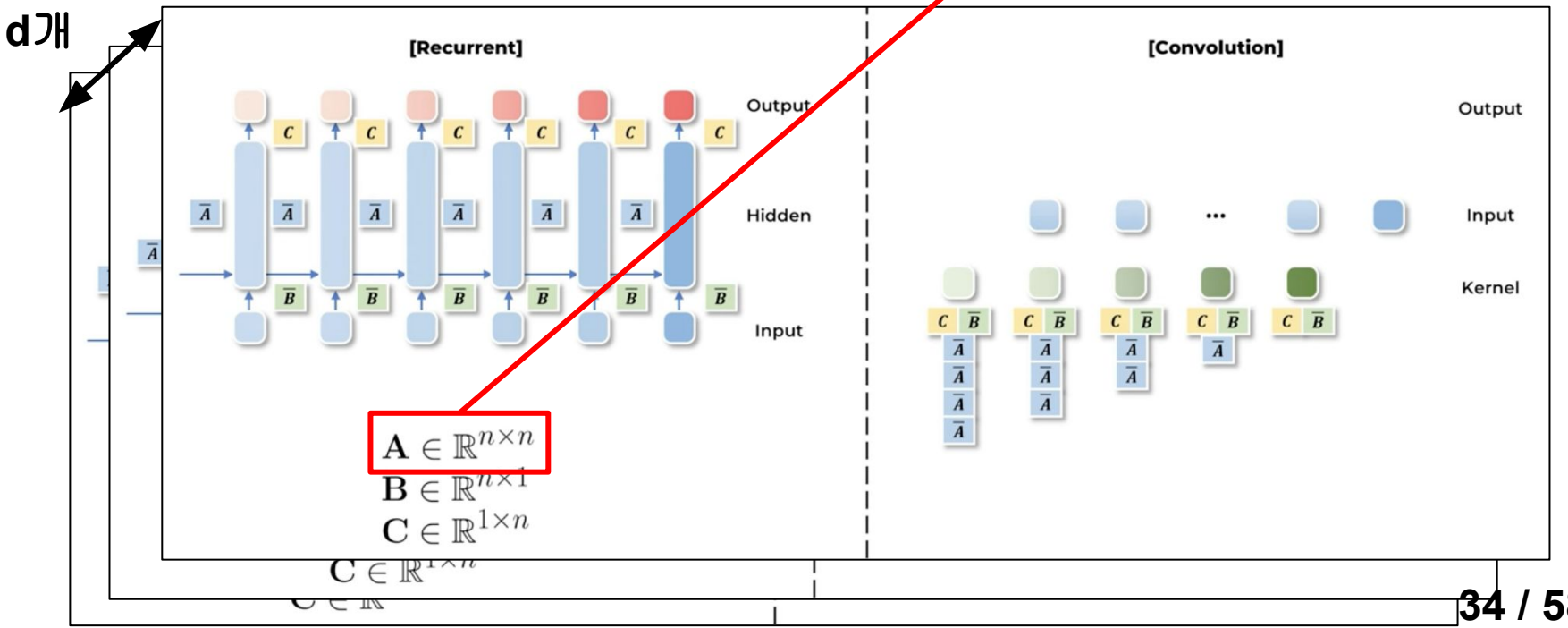
1차원 → **D차원**

*"각 차원 별로 **LSSL**이 **1개씩 독립적**으로 존재한다"*

3. Linear SSM (LSSM)

(6) S4 (Structured State Space Model)

Heavy computation!



3. Linear SSM (LSSM)

(6) S4 (Structured State Space Model)

Diagonalize!

S4(D)

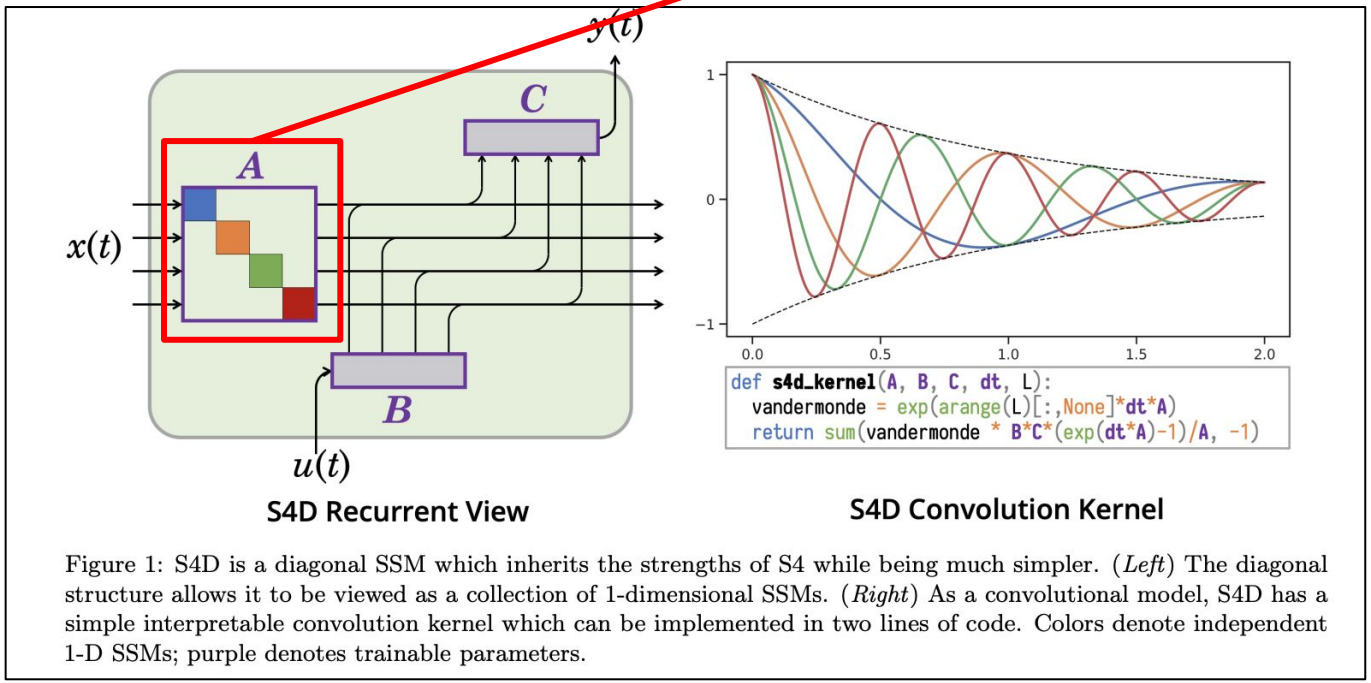
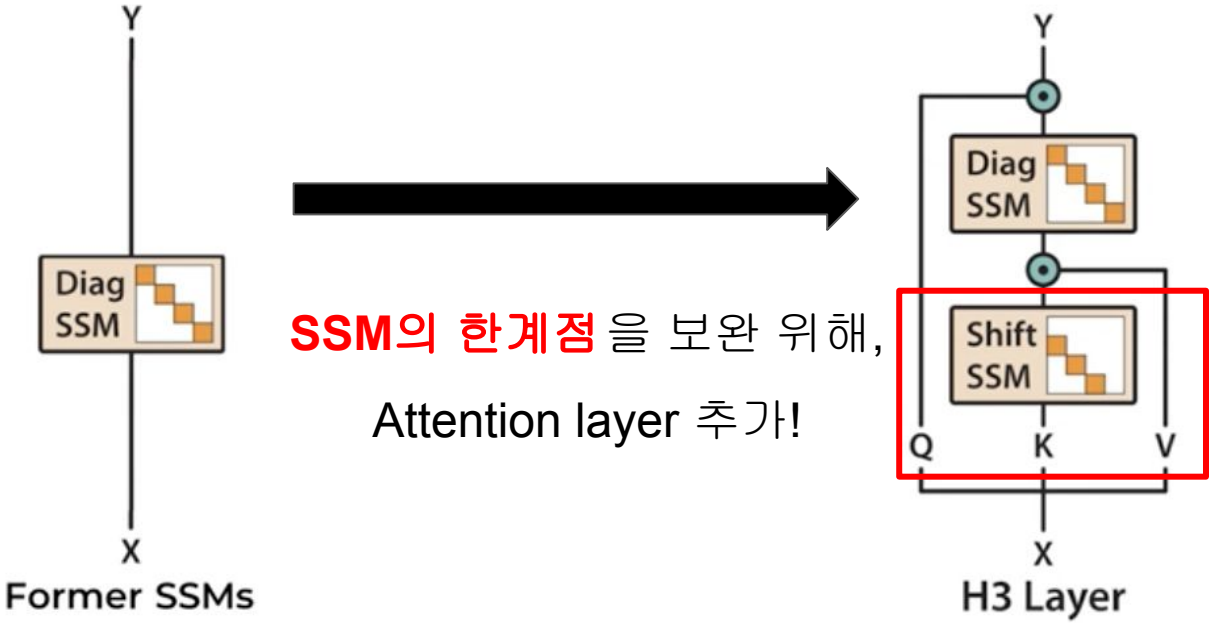


Figure 1: S4D is a diagonal SSM which inherits the strengths of S4 while being much simpler. (*Left*) The diagonal structure allows it to be viewed as a collection of 1-dimensional SSMs. (*Right*) As a convolutional model, S4D has a simple interpretable convolution kernel which can be implemented in two lines of code. Colors denote independent 1-D SSMs; purple denotes trainable parameters.

3. Linear SSM (LSSM)

(7) H3

Hungry Hungry Hippos: Towards Language Modeling with State Space Models



4. MAMBA - Selective SSM

4. MAMBA - Selective SSM

(1) Limitations of S4

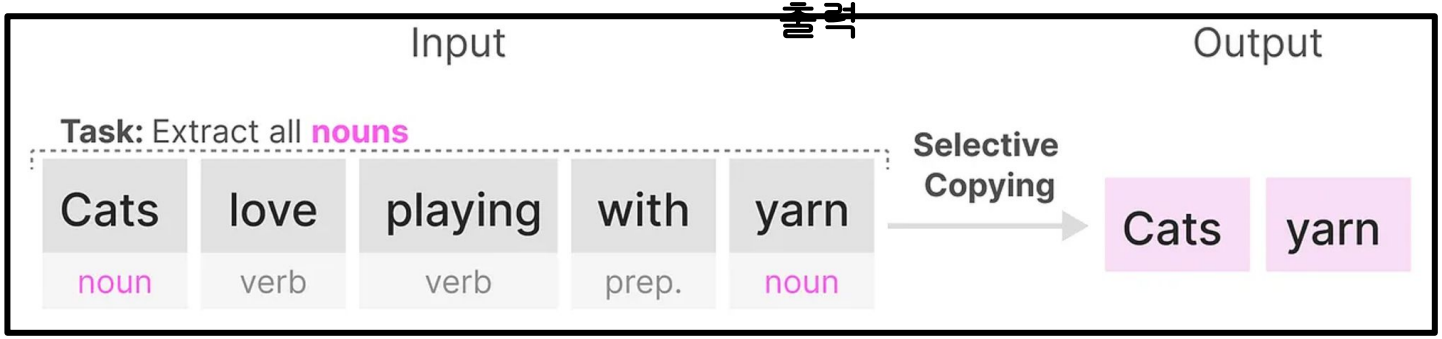
- 한계점: 특정 입력을 선택적으로 집중/무시하는 능력이 떨어짐
- 두 가지 작업
  - (1) 선택적 복사(Selective copying)
  - (2) 유도 헤드(Induction heads)

4. MAMBA - Selective SSM

(1) Limitations of S4

- 한계점: 특정 입력을 선택적으로 집중/무시하는 능력이 떨어짐
- 두 가지 작업
  - (1) 선택적 복사(Selective copying)
  - (2) 유도 헤드(Induction heads)

목표: 입력의 일부를 복사하여 순서대로 출력

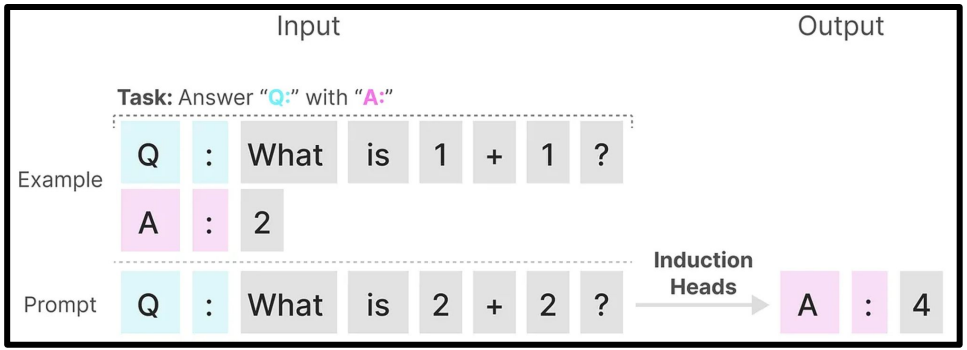


4. MAMBA - Selective SSM

(1) Limitations of S4

- 한계점: 특정 입력을 선택적으로 집중/무시하는 능력이 떨어짐
- 두 가지 작업
  - (1) 선택적 복사(Selective copying)
  - (2) 유도 헤드(Induction heads)

목표: 입력에서 발견된 패턴 재현



4. MAMBA - Selective SSM

(1) Limitations of S4

- 한계점: 특정 입력을 선택적으로 집중/무시하는 능력이 떨어짐
- 두 가지 작업
  - (1) 선택적 복사(Selective copying)
  - (2) 유도 헤드(Induction heads)

실패 이유? **선형 시간 불변성 (Linear Time Invariance, LTI)**

- 행렬 A, B, C는 모든 토큰에 대해 동일 (input INDEPENDENT)

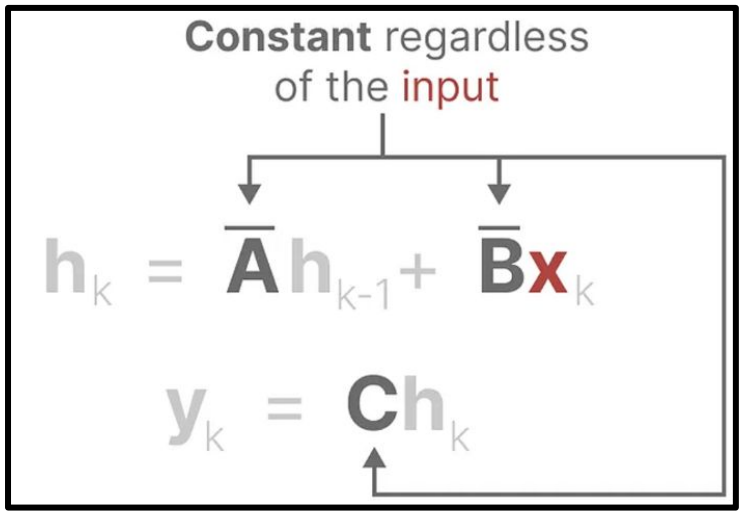
=> 내용 인식 추론 / 기억 불가 (각 토큰을 동등하게 취급하기 때문)

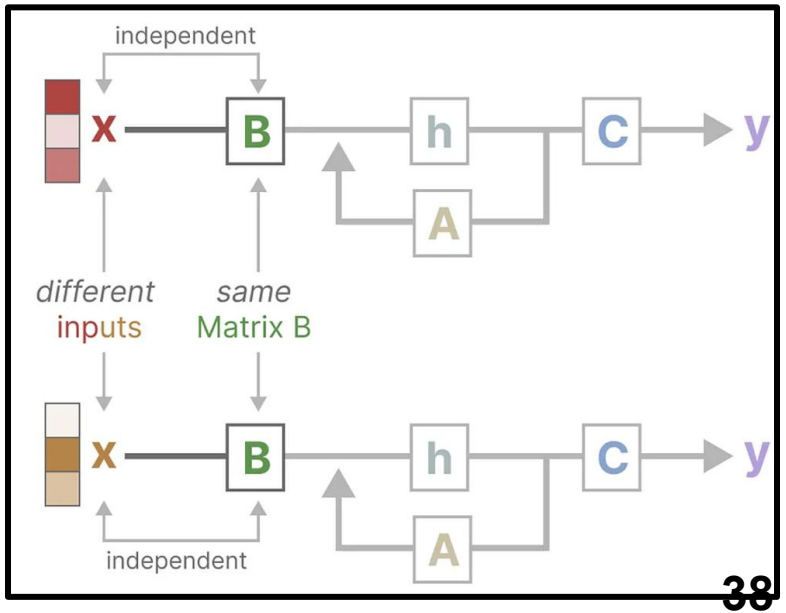
4. MAMBA - Selective SSM

(1) Limitations of S4

선형 시간 불변성 (Linear Time Invariance, LTI)

- 입력(X)에 독립적인 행렬 A,B,C





이에 반해, **Transformer**(의 attention)는 **입력(X)에 의존하는 행렬**을 생성

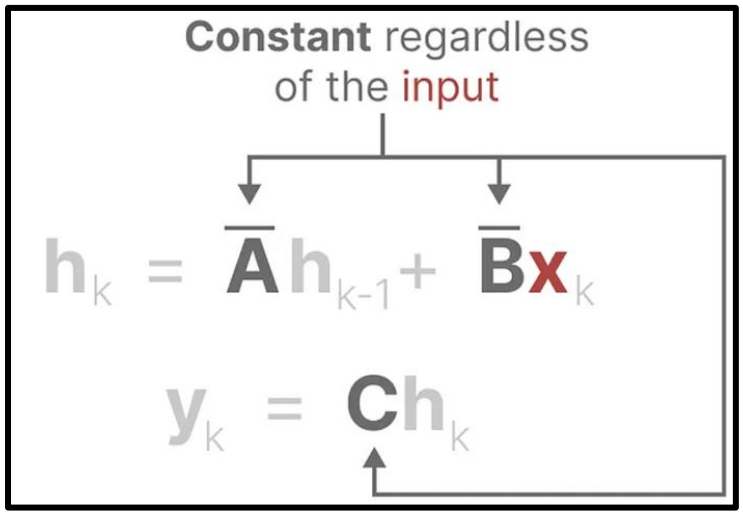
=> 입력에 따라 **선택적**으로 참조!

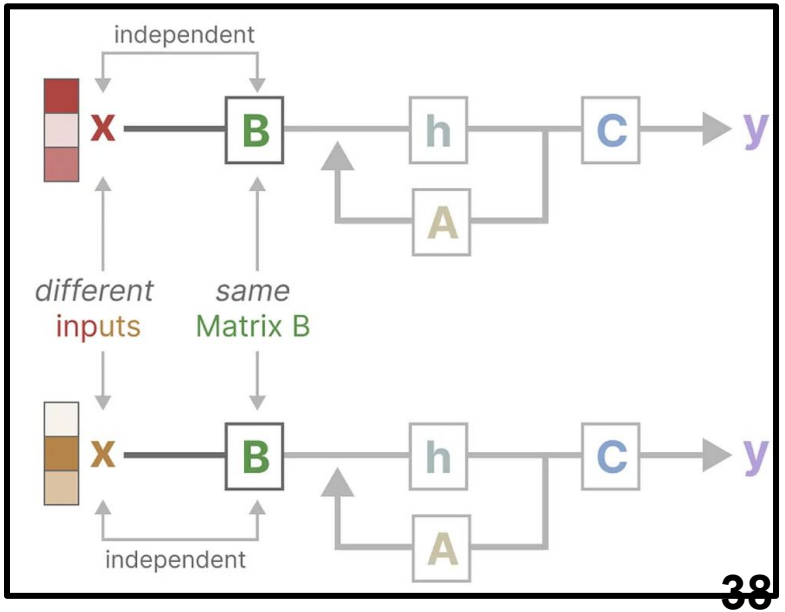
4. M

(1) Li

선형 시간 불변성 (Linear Ti e Invariance, LTI)

- 입력(X)에 독립적인 행렬 ,B,C

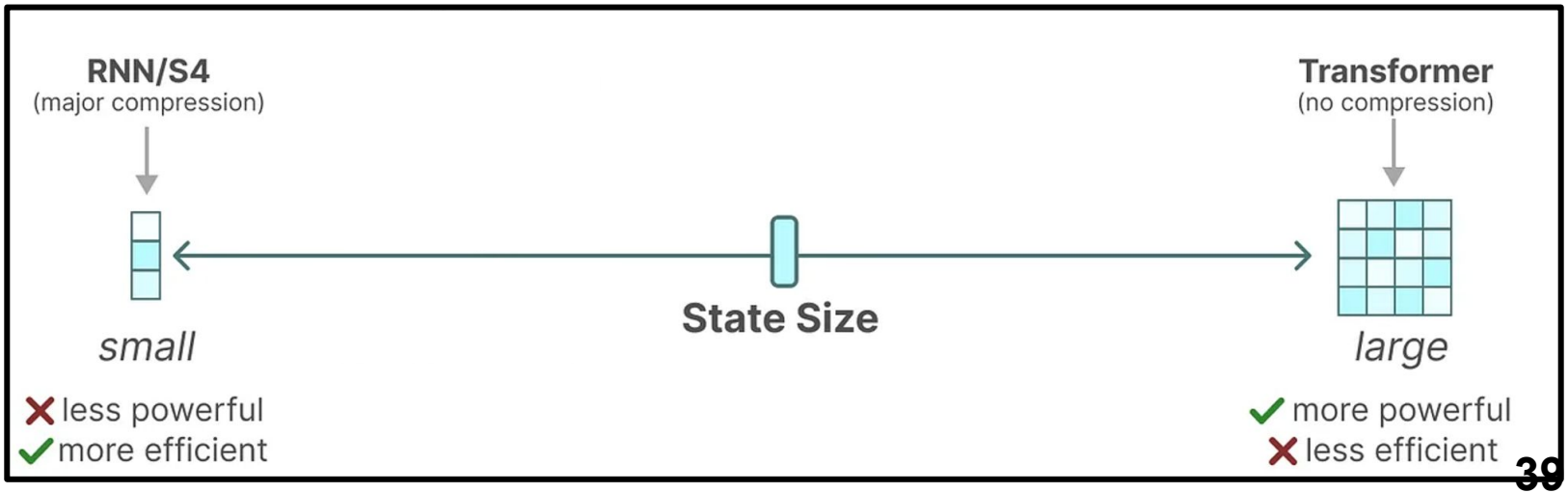




4. MAMBA - Selective SSM

(2) SSM vs. Transformer

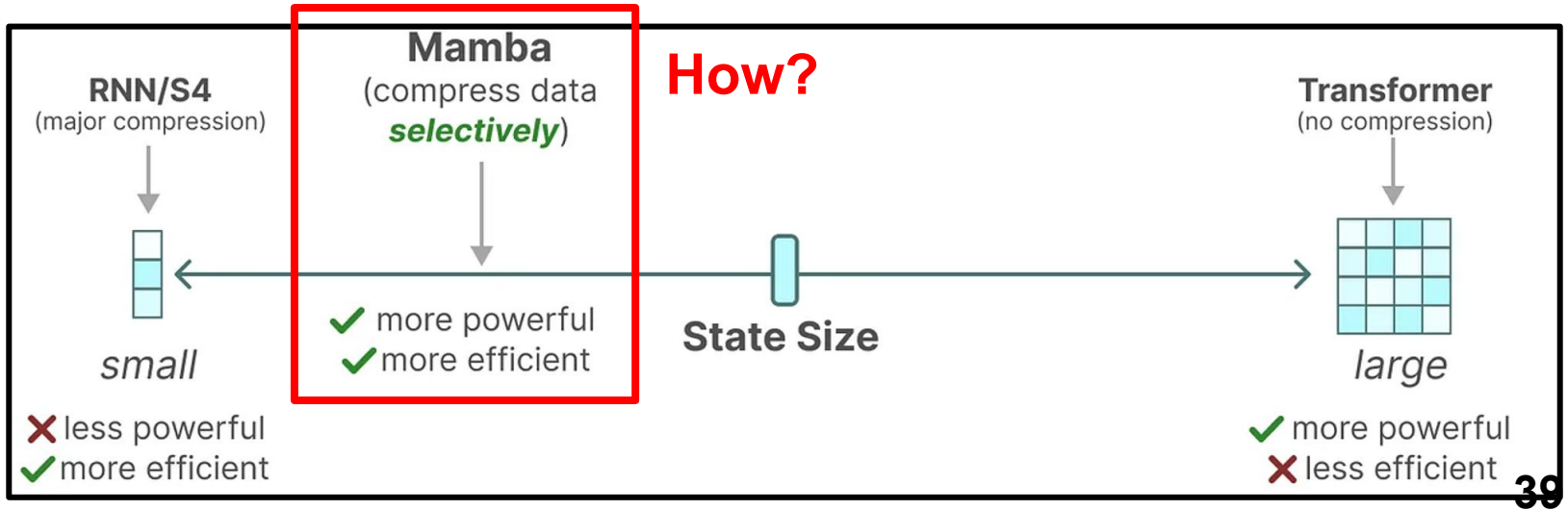
- **SSM**: 압축된(효율적) 표현 + 정적 행렬
- **Transformer**: 덜 압축된(비효율적) 표현 + 동적 행렬



4. MAMBA - Selective SSM

(2) SSM vs. Transformer

- **SSM**: 압축된(효율적) 표현 + 정적 행렬
- **Transformer**: 덜 압축된(비효율적) 표현 + 동적 행렬



4. MAMBA - Selective SSM

(3) MAMBA

SSM & Transformer의 장점을 모두 취한 모형

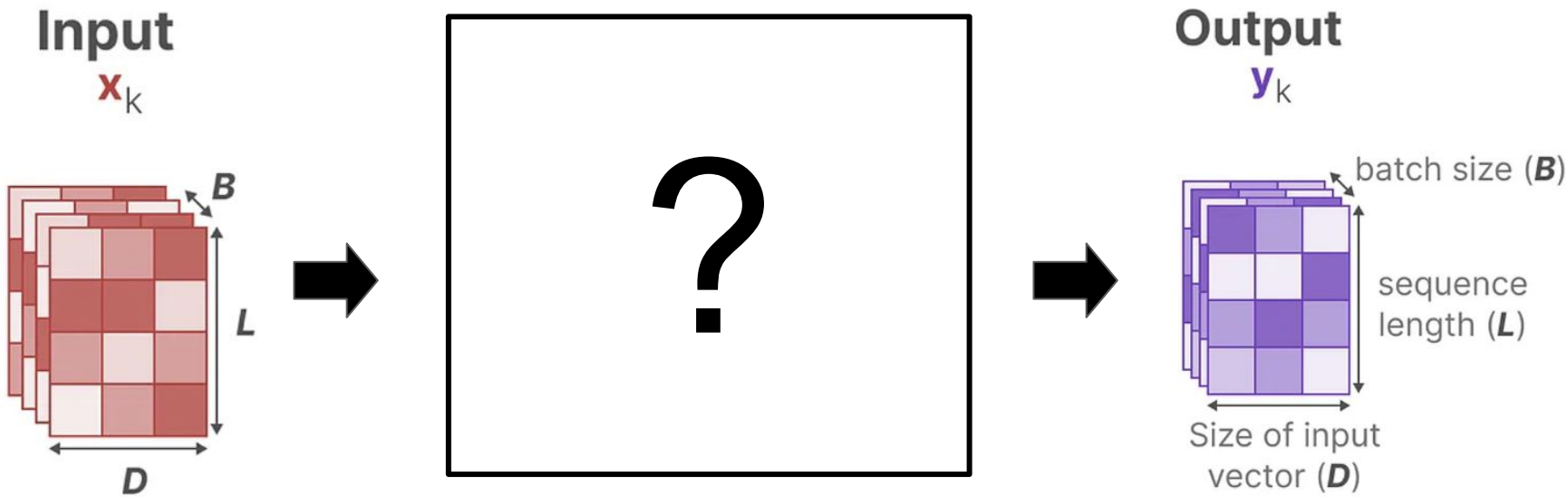
- **SSM**) SSM 구조이므로, “압축된 표현” 가능!
- **Transformer**) ???

Transformer의 장점인 “동적 표현"을 위해, 입력 (X)에 따라 매개변수(A,B,C)가 달라져야!

4. MAMBA - Selective SSM

(3) MAMBA

정적 vs 동적 SSM

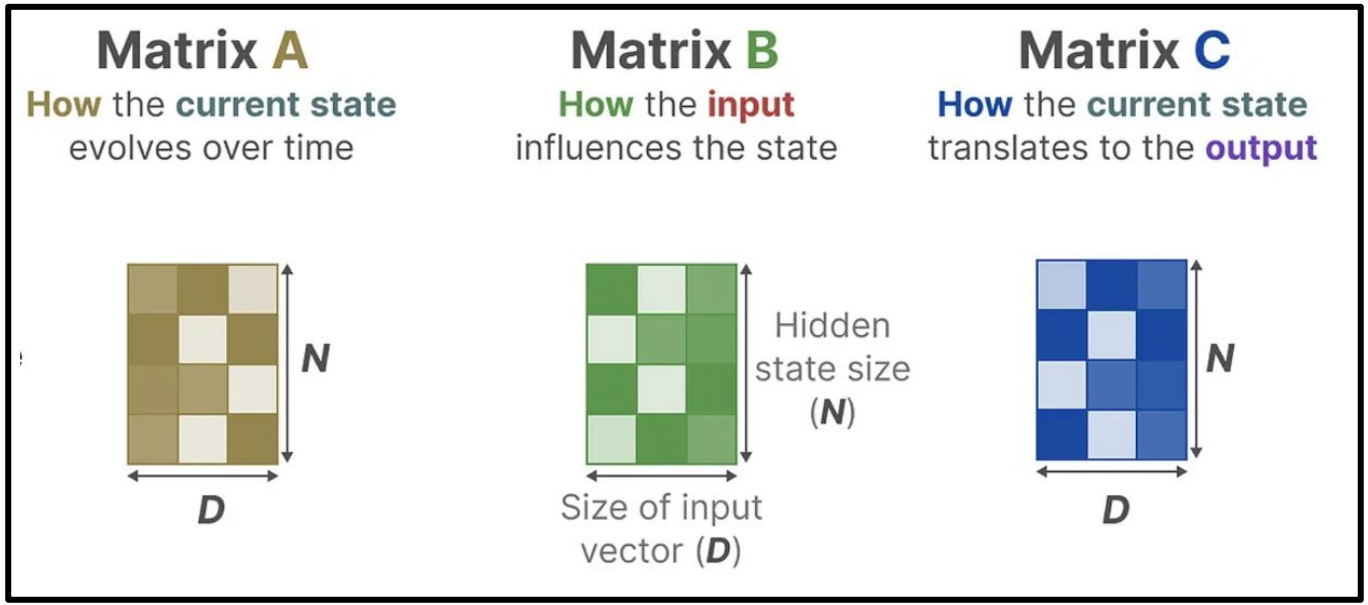


4. MAMBA - Selective SSM

(3) MAMBA

정적인 **SSM: S4 (Structured SSM)**

입력(X)에 대해 같은 **A, B, C** 행렬

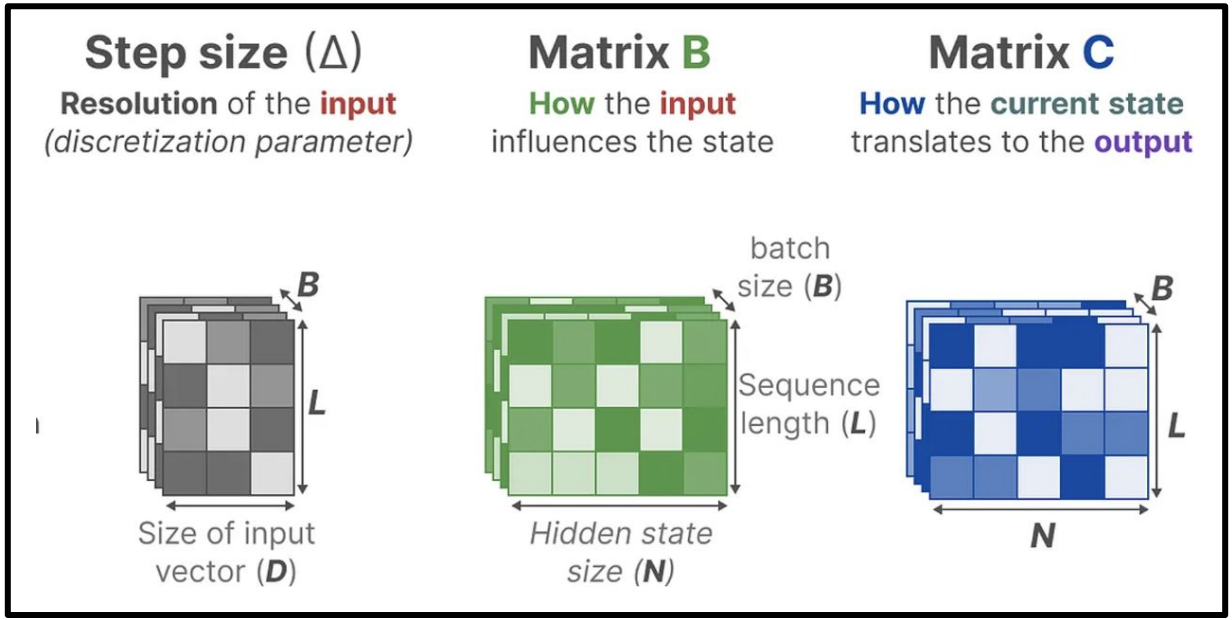


4. MAMBA - Selective SSM

(3) MAMBA

동적인 **SSM: Selective SSM (MAMBA)**

입력(**X**)에 대해 다른 **B, C** 행렬

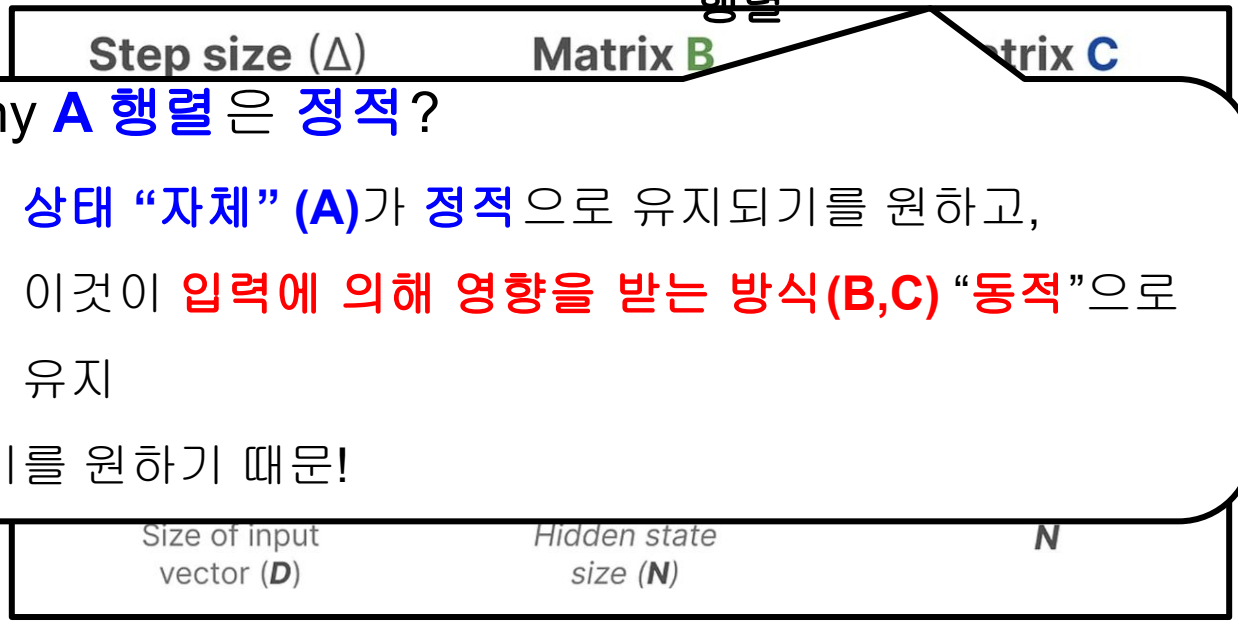


4. MAMBA - Selective SSM

(3) MAMBA

동적인 **SSM: Selective SSM (MAMBA)**

입력(X)에 대해 서로 다른 B, C 행렬



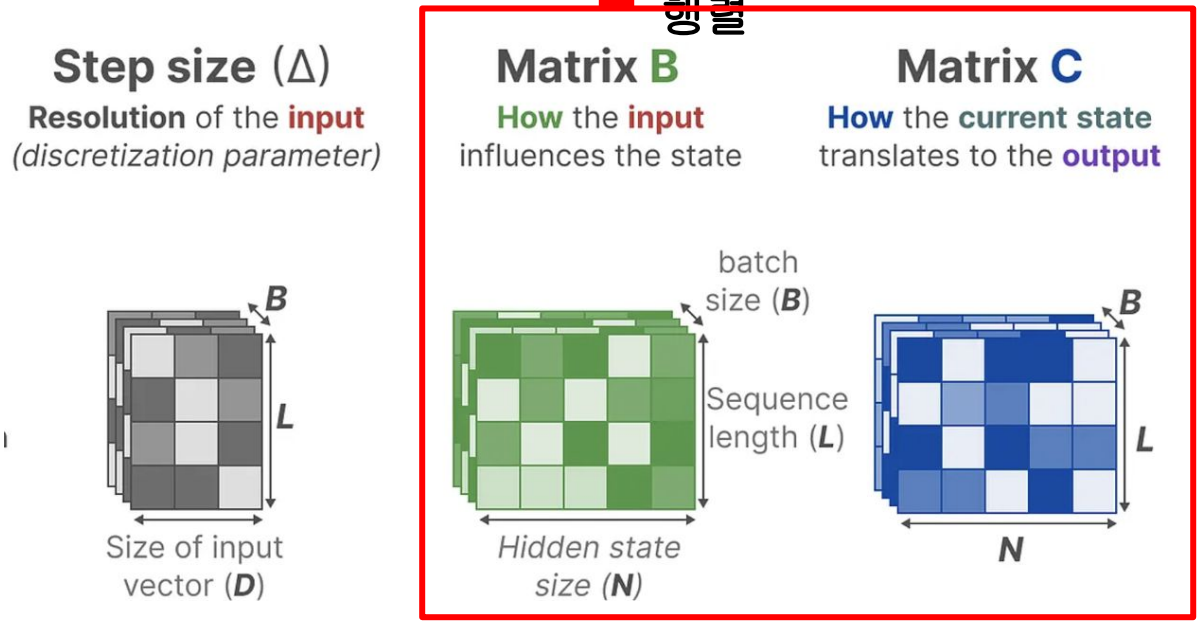
4. MAMBA - Selective SSM

(3) MAMBA

입력에 의존하여 상태에서 무엇을 유지/무시할지 선택

동적인 **SSM: Selective SSM (MAMBA)**

입력(**X**)에 대해 서로 다른 **B, C** 행렬



4. MAMBA - Selective SSM

(3) MAMBA

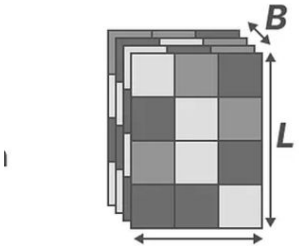
특정 단어를 무시 & 이전 맥락을 더 집중하는 데 초점

동적인 **SSM: Selective SS (MAMBA)**

입력(X)에 대해 서로 다른 **B, C** 행렬

Step size (Δ)

Resolution of the **input**
(discretization parameter)

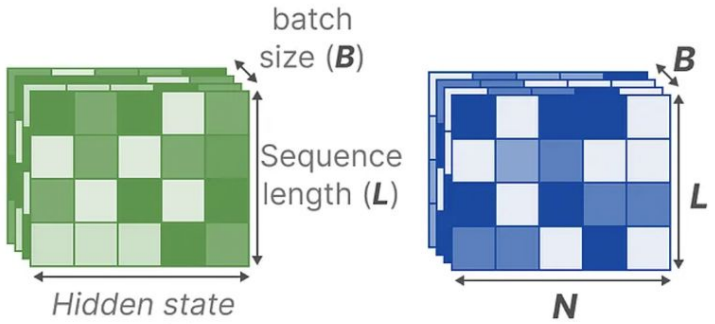


Matrix B

How the **input** influences the state

Matrix C

How the **current state** translates to the **output**



4. MAMBA - Selective SSM

(3) MAMBA

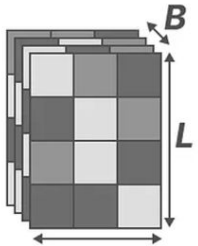
특정 단어를 무시 & 이전 맥락을 더 집중하는 데 초점

동적인 **SSM: Selective SS (MAMB**

Step size (Δ)

Resolution of the **input**

(discretization parameter)



Size of input vector (**D**)

infl

*Hidden state size (**N**)*

N

ZOH Discretization

$$\overline{A} = \exp(\Delta A)$$

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

Δ → 0:
현재 Hidden State 유지
현재 Input 무시

Δ → ∞:
현재 Input에 대한
영향 증가

4. MAMBA - Selective SSM

(3) MAMBA

Algorithm 1 SSM (S4)

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

Input Independent

Algorithm 2 SSM + Selection (S6)

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

Input Dependent
(= Selective)

4. MAMBA - Selective SSM

(4) Two main contributions of MAMBA

1) 선택적 스캔 알고리즘
   - 필요/불필요 정보 선택적 필터링
2) 하드웨어 인식 알고리즘
   - 중간 결과의 효율적인 저장

4. MAMBA - Selective SSM

(5) Contribution 1: 선택적 스캔 알고리즘

스캔 작업

- 행렬: 정적 -> 동적

  => 따라서, convolution 표현 사용 불가! (only 고정된 kernel만 가능하므로)

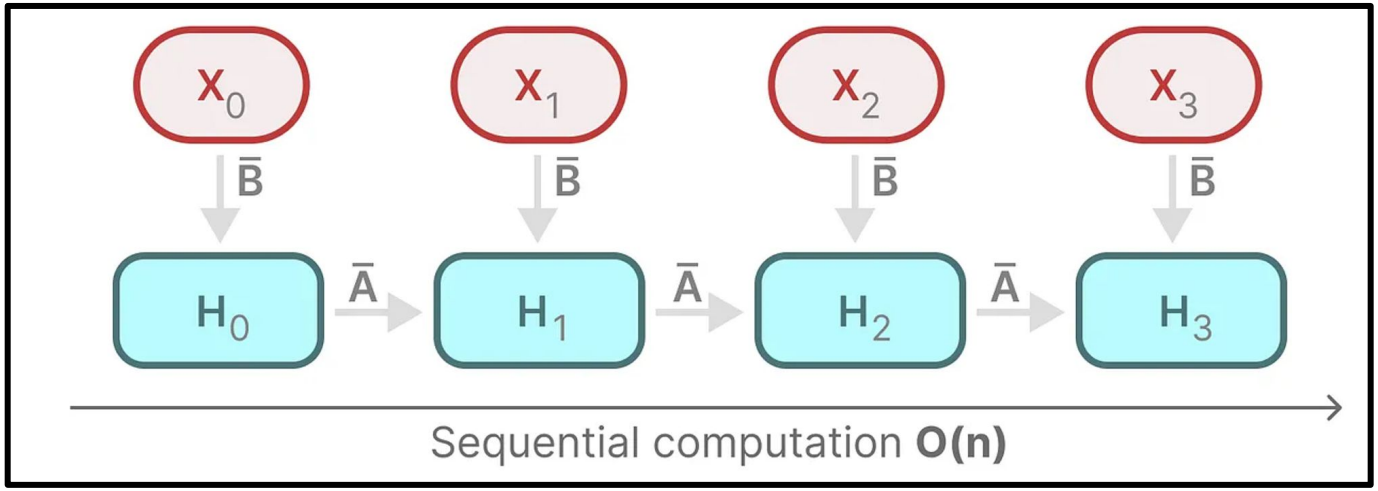
병렬화 불가? NO! … Then, How?

4. MAMBA - Selective SSM

(5) Contribution 1: 선택적 스캔 알고리즘

스캔 작업

- 재귀적 출력



4. MAMBA - Selecti

(5) Contribution 1: 선택

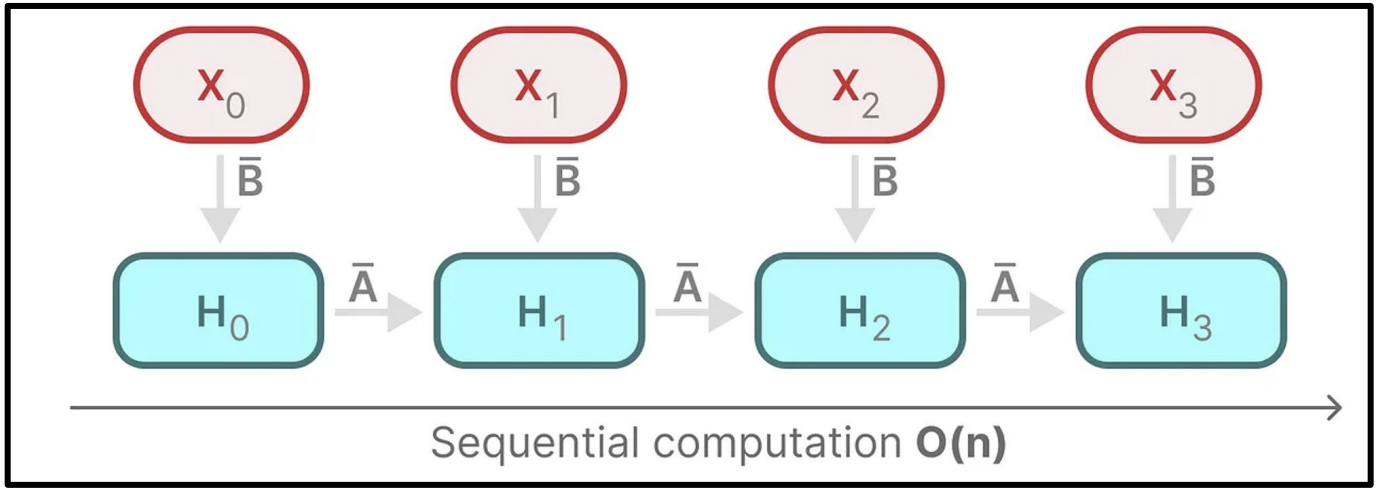
다음 상태 계산 위해, 이전 상태가 필요!

=> 하지만, MAMBA는 **병렬 스캔 알고리즘**을 통해 병렬화 가능!

- 연산 순서가 중요하지 않다고 가정!

스캔 작업

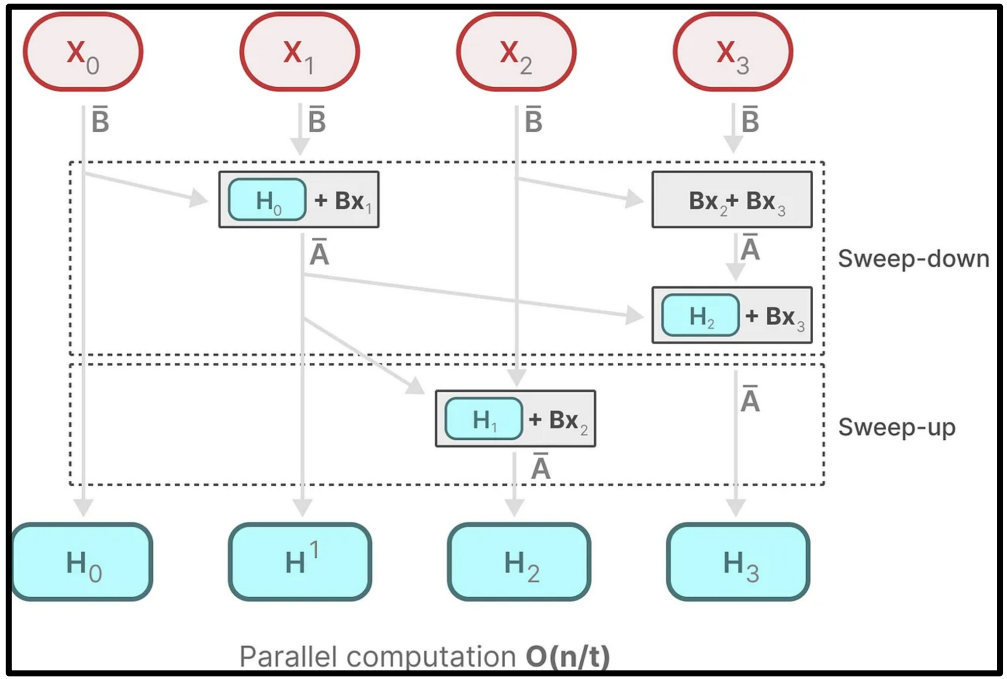
- 재귀적 출력
- 시퀀스를 부분적으로 계산 & 반복적으로 결합



4. MAMBA - Selective SSM

(5) Contribution 1: 선택적 스캔 알고리즘

병렬 스캔 알고리즘



https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda

4. MAMBA - Selective SSM

(5) Contribution 1: 선택적 스캔 알고리즘

연산의 **Associativity** (A->B->C = A->C->B)

먼저 계산할 수 있는 것은 먼저 계산해놓자!

$$
\begin{aligned}
\mathbf{H}_0 &= \overline{\mathbf{B}}_0\mathbf{x}_0 \\
\mathbf{H}_1 &= \overline{\mathbf{A}}_1(\overline{\mathbf{B}}_0\mathbf{x}_0) + \overline{\mathbf{B}}_1\mathbf{x}_1 \\
&= \overline{\mathbf{A}}_1\overline{\mathbf{B}}_0\mathbf{x}_0 + \overline{\mathbf{B}}_1\mathbf{x}_1 \\
\mathbf{H}_2 &= \overline{\mathbf{A}}_2(\overline{\mathbf{A}}_1\overline{\mathbf{B}}_0\mathbf{x}_0 + \overline{\mathbf{B}}_1\mathbf{x}_1) + \overline{\mathbf{B}}_2\mathbf{x}_2 \\
&= \overline{\mathbf{A}}_2\overline{\mathbf{A}}_1\overline{\mathbf{B}}_0\mathbf{x}_0 + \overline{\mathbf{A}}_2\overline{\mathbf{B}}_1\mathbf{x}_1 + \overline{\mathbf{B}}_2\mathbf{x}_2 \\
\mathbf{H}_3 &= \overline{\mathbf{A}}_3(\overline{\mathbf{A}}_2\overline{\mathbf{A}}_1\overline{\mathbf{B}}_0\mathbf{x}_0 + \overline{\mathbf{A}}_2\overline{\mathbf{B}}_1\mathbf{x}_1 + \overline{\mathbf{B}}_2\mathbf{x}_2) + \overline{\mathbf{B}}_3\mathbf{x}_3 \\
&= \overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{A}}_1\overline{\mathbf{B}}_0\mathbf{x}_0 + \overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{B}}_1\mathbf{x}_1 + \overline{\mathbf{A}}_3\overline{\mathbf{B}}_2\mathbf{x}_2 + \overline{\mathbf{B}}_3\mathbf{x}_3
\end{aligned}
$$

Compute Matrices Multiplication with Multiple Threads

Complexity Drops to $O(n/T)$

4. MAMBA - Selective SSM

(5) Contribution 2: 하드웨어 인식 알고리즘

앞서 Time Complexity는 해결했으나.. **Memory Complexity** issue!!

- Transformer: (B,L,D)
- MAMBA: (B,L,D,**N**)

4. MAMBA - Selective SSM

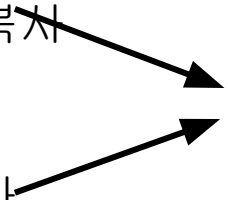
전송 속도 비교

- **SRAM**: 작지만 효율적
- **DRAM (HBM)**: 크지만 비효율적

(5) Contribution 2: 하드웨어 인식 알고리즘

(Matrix Multiplication을 제외한) 대부분의 GPU 연산:

- 순서 1) **DRAM**에서 **SRAM**으로 텐서를 복사
- 순서 2) **SRAM**에서 작업
- 순서 3) **SRAM**에서 **DRAM**으로 다시 복사



많은 시간 소요

(compared to step 2)

Solution??

4. MAMBA - Selective SSM

전송 속도 비교

- **SRAM**: 작지만 효율적
- **DRAM (HBM)**: 크지만 비효율적

(5) Contribution 2: 하드웨어 인식 알고리즘

Step 1) 복사의 부담을 줄이기 위해,

- **4개의 dimension**이 아니라
- **3개의 dimension**만을 복사하여

SRAM으로 보낸다!

Step 2) Discretization (3->4차원 과정)을 **DRAM**이 아니라 **SRAM**에서 진행

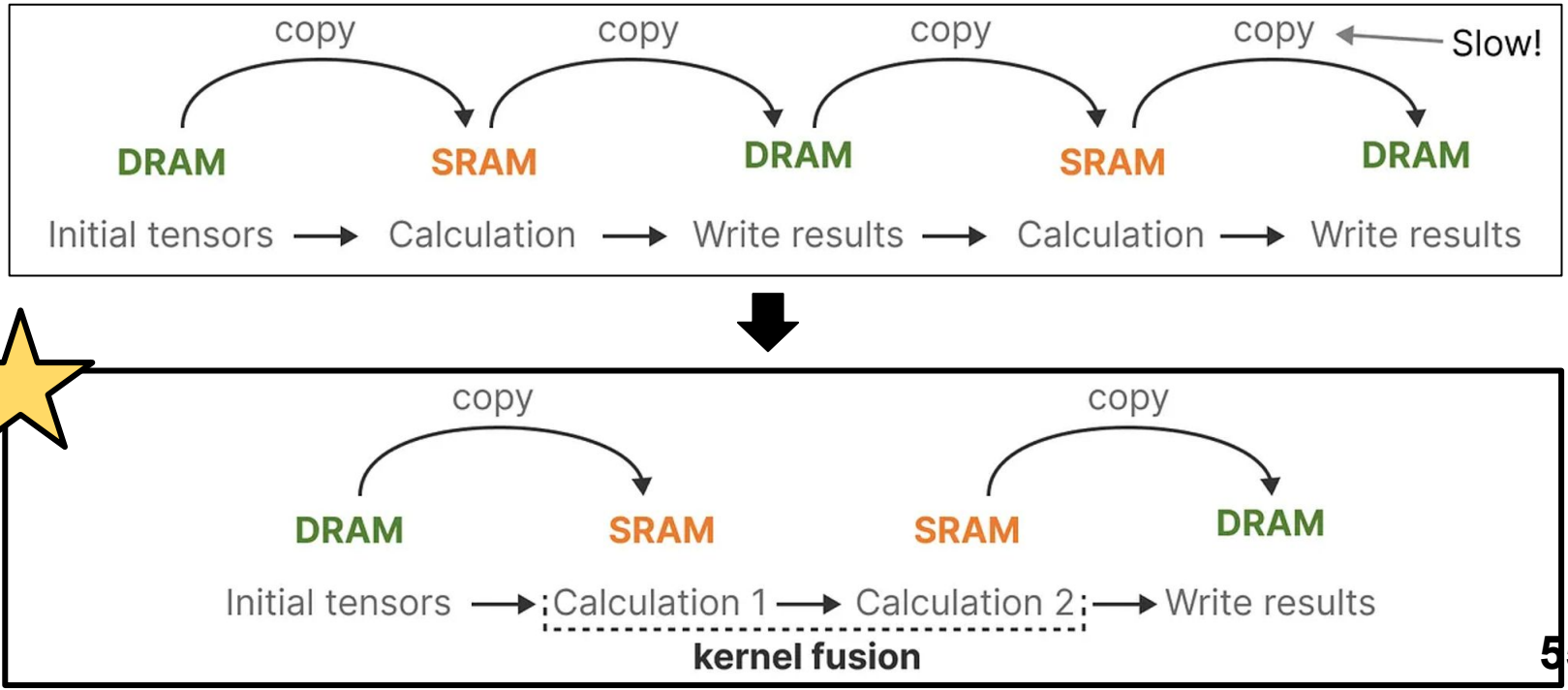
Step 3) (작업 완료 후, 4->3차원이 된) 결과물 y를 **SRAM**에서 **DRAM**으로 복사

4. MAMBA - Selective SSM

전송 속도 비교

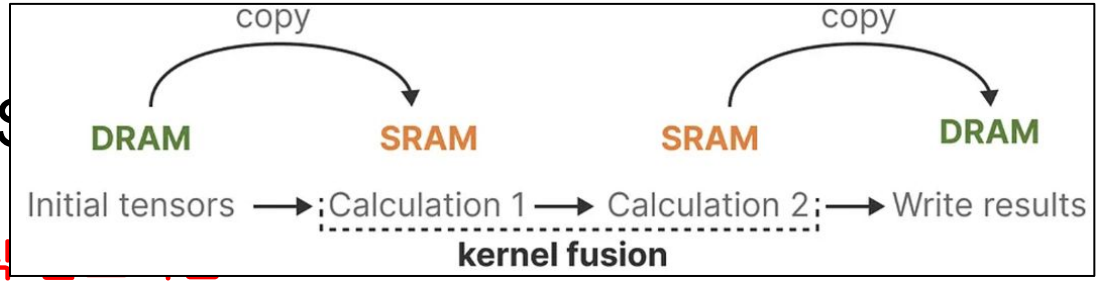
- **SRAM**: 작지만 효율적
- **DRAM**: 크지만 비효율적

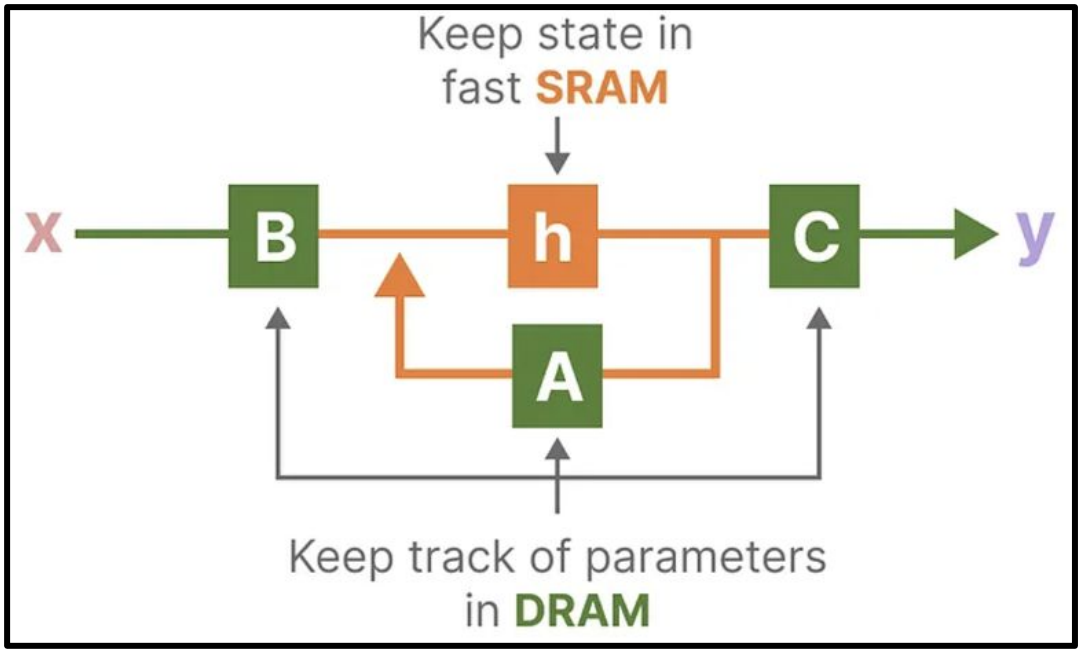
(5) Contribution 2: 하드웨어 인식 알고리즘



4. MAMBA - Selective S

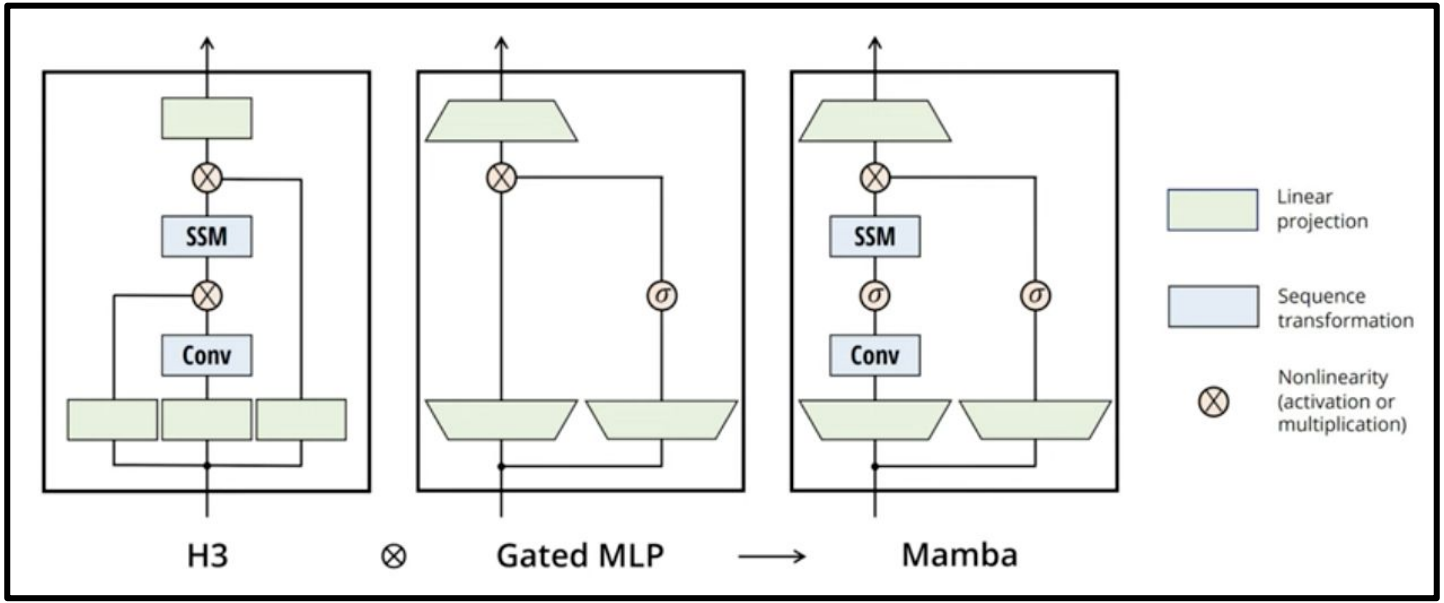
(5) Contribution 2: 하드웨어 인식





4. MAMBA - Selective SSM

(6) Architecture



5. Conclusion

MAMBA = Selective SSM

- Fast Training + Fast Inference
- Selective Algorithm (Input dependent)

References

- A. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” arXiv preprint arXiv:2312.00752, 2023.
- https://youtu.be/JjxBNBzDbNk
- https://tulip-phalange-a1e.notion.site/05f977226a0e44c6b35ed9bfe0076839