

CH16. 새로운 아키텍처

시계열 모델은 RNN → LSTM → Transformer → S4 → Mamba로 진화 중

□ 기존 아키텍처의 장단점

□ **SSM** (State Space Model, 상태(state)를 기반으로 시간 흐름을 모델링하는 프레임워크, RNN, LSTM 등)

: **S4 알고리즘** (긴 시퀀스에 대한 추론 성능과 계산 효율성 모두를 달성한 모델)

□ 선택 메커니즘

□ **맘바** (선택 메커니즘을 활용한 새로운 SSM 계열 모델, 연산 효율성과 정확도 동시 확보)

발표에 앞서

RNN → LSTM → Transformer → S4 → Mamba로 진화 중

- RNN : 시계열 데이터나 순차 데이터를 처리하는 최초의 딥러닝 구조 (but 장기 의존성 문제)
- LSTM : 셀 상태와 게이트 구조를 사용하여 필요한 정보 기억 (but 구조가 복잡함)
- Transformer : Attention 메커니즘으로 전체 입력 시퀀스를 동시에 처리, 병렬 연산 (but 계산량 많음)
- S4 : 기존 RNN 계열과 Transformer의 장점을 절충한 구조 (but)
- Mamba : Transformer보다 가볍고 빠르며, 더 긴 시퀀스도 잘 처리

History

대표 모델	핵심 기술	장점	한계
RNN	순환 구조	시간 흐름 반영	장기 의존성 불안정
LSTM	게이팅 구조	장기 정보 유지	계산 복잡
Transformer	Attention	병렬화, 정확도 ↑	메모리 과다
S4	State Space + FFT	긴 시퀀스 처리	구조 이해 어려움
Mamba	선택 메커니즘	빠름 + 효율적	최신 구조, 연구 중

HOW



기존 아키텍처의 장단점

□ 트랜스포머 vs RNN

- 긴 문장도 성능 유지, 병렬 연산 가능
- 시퀀스 길이 제공에 비례 (연산량)
→ 추론 시간 개선 필요

	RNN	트랜스포머
장점	<ul style="list-style-type: none">• 추론이 효율적임(시퀀스 길이에 관계없이 토큰 당 생성시간이 일정함)	<ul style="list-style-type: none">• 어텐션을 활용하면서 시퀀스 길이가 길어져도 성능이 유지• RNN 대비 성능이 높음• 학습 시 병렬 연산 가능
단점	<ul style="list-style-type: none">• 학습 시 병렬 연산이 어려움• 그레이디언트 소실 같은 문제로 학습 불안정• 시퀀스 길이가 길어지면 성능이 하락	<ul style="list-style-type: none">• 학습 시 시퀀스 길이에 제곱에 비례해서 학습 시간 증가• 추론 시 시퀀스 길이에 비례하게 추론 시간 증가

□ 목표

- 트랜스포머보다 연산이 가벼우면서 성능이 높은 모델을 개발한다.
- SSM 계열의 추론 효율성을 유지하면서 병렬연산이 가능할 수 있을까?

□ SSM

- SSM은 내부상태를 가지고, 시간에 따라 달라지는 시스템을 해석하는 모델링 방법
- 최종 목표인 맘바도 SSM 계열임

□ 식

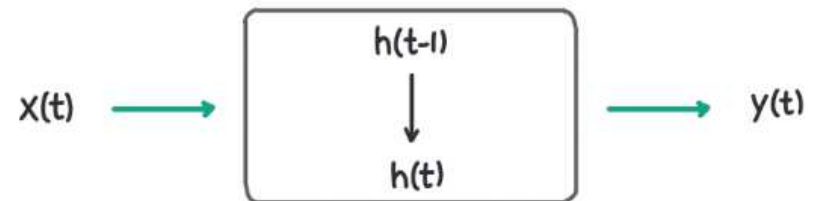
$$h(t) = \mathbf{A}h(t-1) + \mathbf{B}x(t)$$

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

h : 모델의 내부 상태

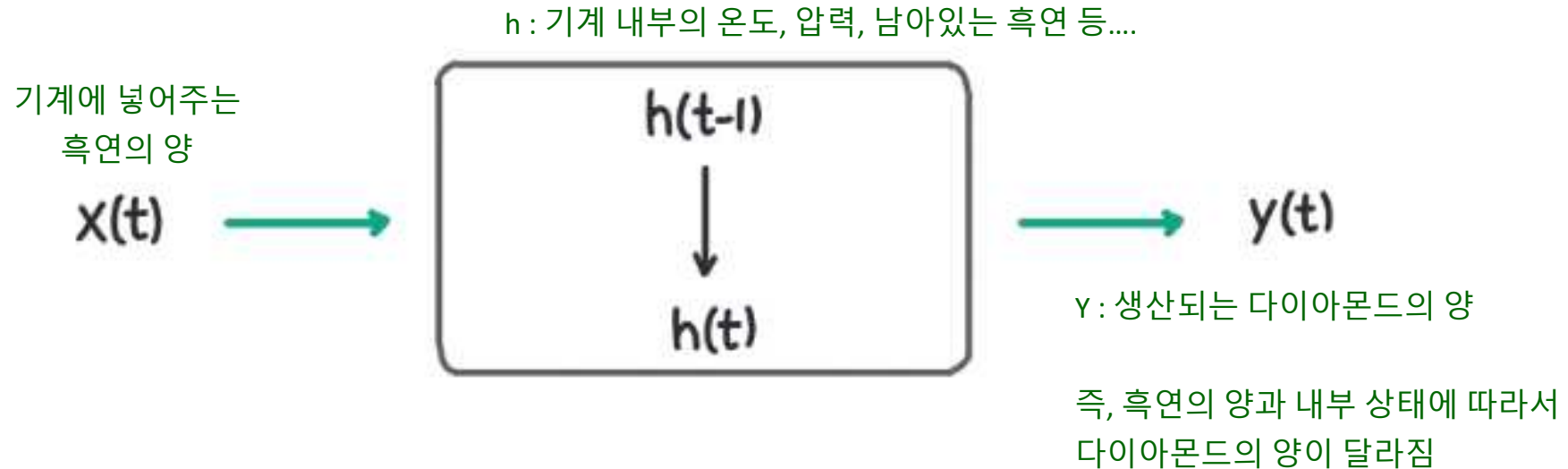
x : input값

y : 출력



- t 시점에 입력이 들어오면 내부상태 h 가 변경되고, 그에 준하는 y 가 도출됨

다이아몬드를 만들자



$$h(t) = Ah(t-1) + Bx(t)$$

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

A,B,C,D 모두 행렬이므로 선형 관계를 가정한다.

딥러닝과 같이 비선형 관계 연산이 없음

→ SSM은 비선형 연산을 제거해서 계산 효율성을 높였다.

$$: h(t) = \tanh(Ah(t-1) + Bx(t))$$

비선형(예: 곱하기, 제곱, 시그모이드 등)

S4 알고리즘

□ S4

- S4는 효율적인 학습을 위해서 A, B, C, D가 시간에 따라 변하지 않도록 고정함
(선형시간 불변성 특성)

$$h(0) = \mathbf{B}x(0)$$

$$h(1) = \mathbf{A}h(0) + \mathbf{B}x(1) = \mathbf{A}\mathbf{B}x(0) + \mathbf{B}x(1)$$

$$h(2) = \mathbf{A}h(1) + \mathbf{B}x(2) = \mathbf{A}^2\mathbf{B}x(0) + \mathbf{A}\mathbf{B}x(1) + \mathbf{B}x(2)$$

$$h(t) = \sum_{k=0}^t \mathbf{A}^{t-k} \mathbf{B}x(k)$$

입력 $x(k)$ 를 가중합해서 누적하는 형태
각 $x(k)$ 는 계수 $\mathbf{A}^{t-k}\mathbf{B}$ 에 의해 필터링되고 더해짐
: 컨볼루션의 정의와 동일

→ A, B를 행렬화하여 한번에 계산

$$h = \begin{bmatrix} \mathbf{B} & 0 & 0 \\ \mathbf{AB} & \mathbf{B} & 0 \\ \mathbf{A}^2\mathbf{B} & \mathbf{AB} & \mathbf{B} \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} = \mathbf{K} \times x$$

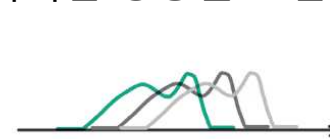
$$y = \mathbf{C} \times h + \mathbf{D} \times x$$

→ 출력 y 는 h 에 고정된 c 를 곱하면 되므로
출력까지 한번에 계산

□ 학습/추론

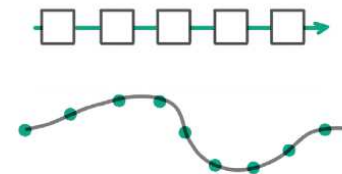
- 학습에서는 컨볼루션 연산을 통해 병렬 계산
여러 시점의 입력 데이터를 처리
- 추론에서는 순차적으로 하나씩 출력을 생성

→ 학습과 추론 모두 효율적이면서 긴 시퀀스 입력에 뛰어난 성능을 보임



$$y = K * x$$

CNN



$$\begin{aligned} h &= \mathbf{A}h + \mathbf{B}x \\ y &= \mathbf{C}h + \mathbf{D}x \end{aligned}$$

RNN

S4D/S5 같은 변형모델도 연구됨

S4 알고리즘



병렬화 가능성 요약

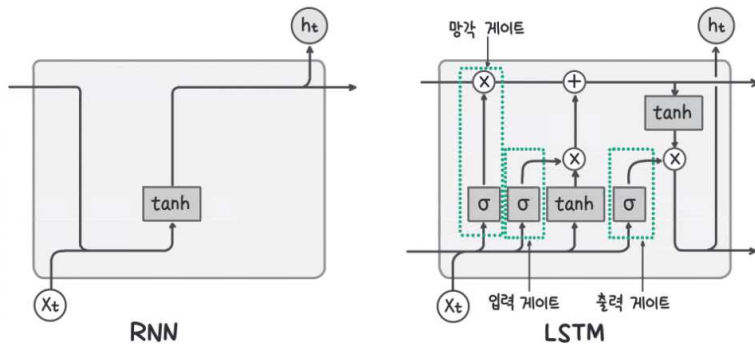
모델	병렬 처리	설명
RNN/LSTM	❌ 불가능	시점 간 의존성이 강해 순차처리
Transformer	✅ 가능	모든 위치에서 self-attention 병렬 가능
S4 (SSM)	✅ 가능	커널 기반 컨볼루션 → FFT → 병렬화 가능

선택 메커니즘

□ RNN 모델의 맥락을 저장하는 제한된 크기의 공간

- 다양한 길이의 입력을 제한된 공간에 저장 → 압축
- RNN : 입력을 h 에 그대로 누적
- LSTM : 맥락을 더 효율적으로 압축하기 위해서

입력/기존 정보를 얼마나 망각할지 게이트 추가



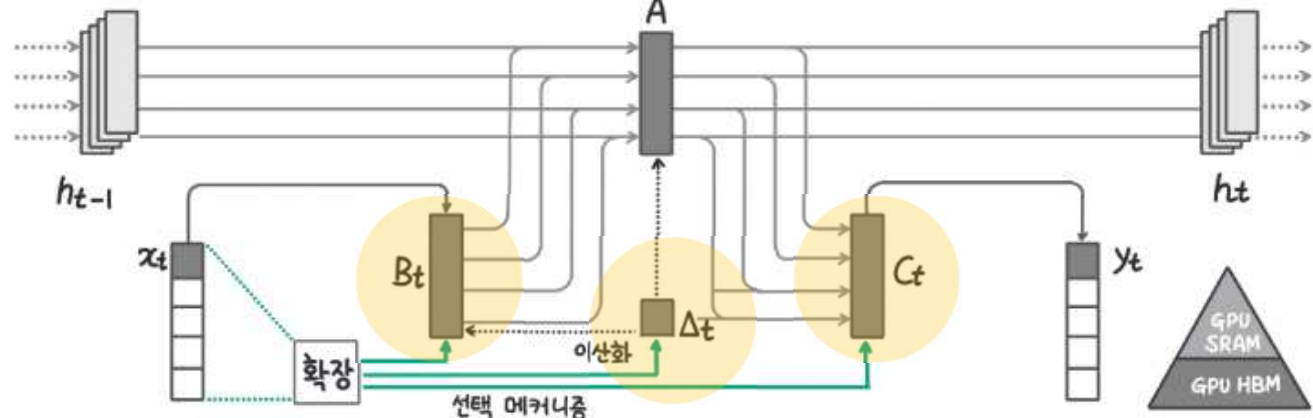
□ S4의 맥락을 압축하는 능력

- S4는 RNN에서 비선형 함수가 사라진 형태
→ 맥락을 압축하는 능력 부족

- 맘바는 S4의 부족한 압축 능력을 보강하기 위해서
선택 메커니즘 추가함 (꼭 필요한 정보만 저장)

맘바의 등장 및 원리

학습 과정에서 어떤 맥락에서 어떤 입력을 저장하는 게 가장 도움이 되는지 판단 (B, C)
→ 기존 방식에 비해 효율적인 압축 가능



Δt : 이전 맥락을 얼마나 업데이트할 지 결정하는 인자
(RNN에서 게이트와 동일한 역할)

선택 메커니즘을 사용하는 mamba의 고민

□ mamba는 입력에 따라 다른 B, C, 델타를 사용

- s4의 학습에 사용하던 컨볼루션 사용 불가
- mamba는 계산 효율을 높여야 함

□ 계산 효율 향상을 위한 방안

① 커널 퓨전 : GPU IO 줄이기

② 중간 결과물 재계산

역전파에 필요한 중간결과물을 저장하지 않고,
필요할 때만 재계산

③ 병렬 스캔

S4

알고리즘 1 S4

입력: $x : (B, L, D)$

출력: $y : (B, L, D)$

1: $A : (D, N) \leftarrow$ 파라미터

▷ 구조화된 $N \times N$ 행렬

2: $B : (D, N) \leftarrow$,라미터

3: $C : (D, N) \leftarrow$,라미터

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

5: $\bar{A}, \bar{B} : (D, N) \leftarrow$ 이산화(Δ, A, B)

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

▷ 시간에 따라 불변: RNN, *NN 연산 가능

7: return y

mamba

알고리즘 2 mamba

입력: $x : (B, L, D)$

출력: $y : (B, L, D)$

1: $A : (D, N) \leftarrow$,라미터

▷ 구조화된 $N \times N$ 행렬

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

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

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

5: $\bar{A}, \bar{B} : (B, L, D, N) \leftarrow$ 이산화(Δ, A, B)

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

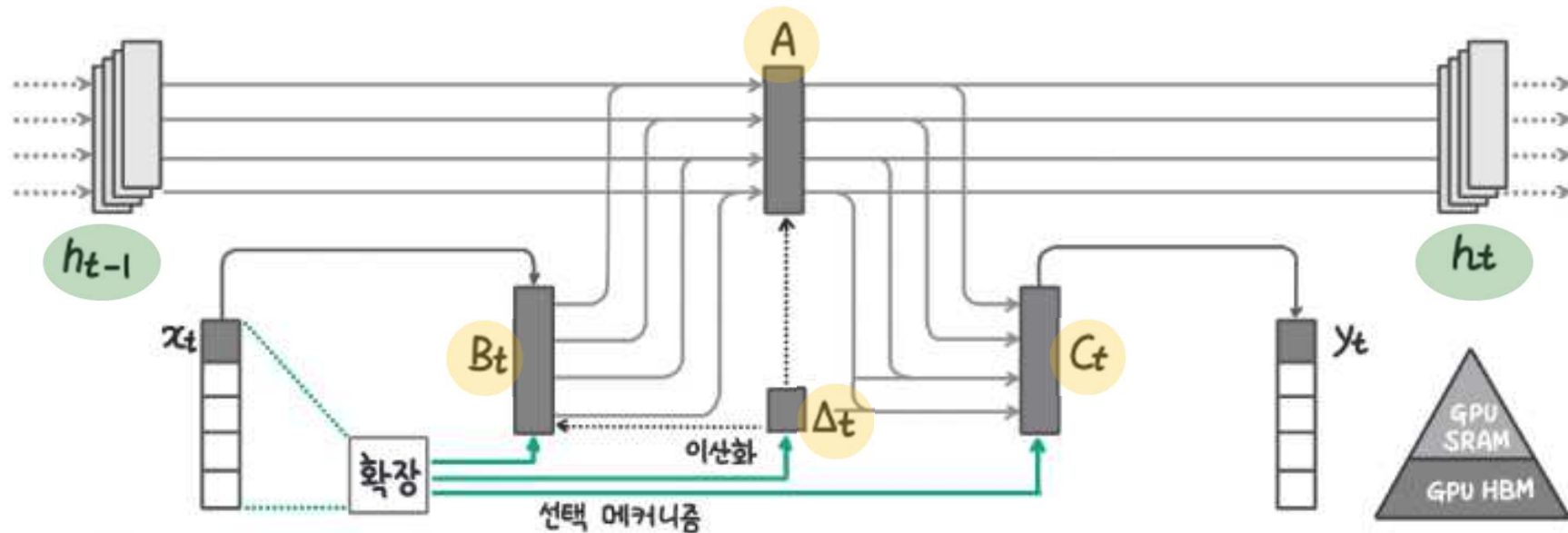
▷ 시간에 따라 변함: 순차 개선헤스캔만 가능

7: return y

선택 메커니즘을 사용하는 맘바의 고민

□ 커널 퓨전 : GPU IO 줄이기

- A, B, C, 델타 가중치는 저장 공간이 가장 큰 GPU 고대역폭 메모리에 저장
 - 계산 중간 과정인 잠재 상태 h 는 크기가 작지만 연산속도가 훨씬 빠른 GPU SRAM에 저장 (외부에 꺼내지 않음)
- GPU IO를 대폭 줄임



□ 중간 결과물 재계산

- 역전파에 필요한 중간 결과물은 역전파 시점에 다시 계산
- : 연산량은 늘어나지만 GPU IO를 줄일 수 있어서 결과적으로 연산 시간 단축 효과 발생

선택 메커니즘을 사용하는 mam바의 고민

항목	S4	Mamba
연산 방식	전체 시퀀스를 컨볼루션	선택적으로 업데이트 (RNN 유사)
중간 상태(h) 저장	GPU SRAM (커널 퓨전)	GPU SRAM 또는 레지스터 (선택적 연산)
파라미터 저장 위치	GPU HBM (A, B, C, Δt)	GPU HBM (필터 파라미터)
GPU IO 줄이는 방법	커널 퓨전 + 병렬 컨볼루션	연산 횟수 자체를 줄임 (selective compute)
병렬화 가능성	높음 (전체 컨볼루션)	제한적 (선택적 업데이트)
연산 효율성	메모리 locality 극대화	연산 스킵으로 FLOPs 감소
어떻게 하면 전체 데이터를 연산하기 위해서 GPU메모리를 잘 활용할까?		어떻게 하면 연산할 데이터를 줄일까?

- S4는 GPU의 **메모리 계층 구조 (HBM ↔ SRAM)**를 잘 활용하여 중간 상태를 GPU 내부에서만 처리하며 병렬 연산 최적화
- Mamba는 병렬보다는 선택적 연산/업데이트를 통해 GPU 자원 소비 자체를 줄이는 방식
- 정확도와 복잡한 구조 모델링이 필요하다면 S4,
- 빠른 응답과 경량화 모델이 필요하다면 Mamba가 적합

선택 메커니즘을 사용하는 망바의 고민

□ 병렬 연산

- 잠재상태 h 의 선형성 (RNN은 비선형성이므로 순차적 연산이 필요함)

$$\text{RNN} \quad h_t = \tanh(x_t W_{ih}^T + b_{ih} + h_{t-1} W_{hh}^T + b_{hh})$$

$$\text{SSM} \quad h_t = \bar{A}h_{t-1} + \bar{B}x_t$$

- 잠재상태 h 의 계산에 필요한 A, B 는 입력 x 에 따라서 달라지지만 입력 x 와 선형적인 관계임
→ 병렬연산이 가능함(s4 보다는 제한적)

$$h(0) = \mathbf{B}_0 x(0)$$

$$h(1) = \mathbf{A}_1 h(0) + \mathbf{B}_1 x(1) = \mathbf{A}_1 \mathbf{B}_0 x(0) + \mathbf{B}_1 x(1)$$

$$h(2) = \mathbf{A}_2 h(1) + \mathbf{B}_2 x(2) = \mathbf{A}_2 \mathbf{A}_1 \mathbf{B}_0 x(0) + \mathbf{A}_2 \mathbf{B}_1 x(1) + \mathbf{B}_2 x(2)$$

s4의 병렬연산

$$h(0) = \mathbf{B}x(0)$$

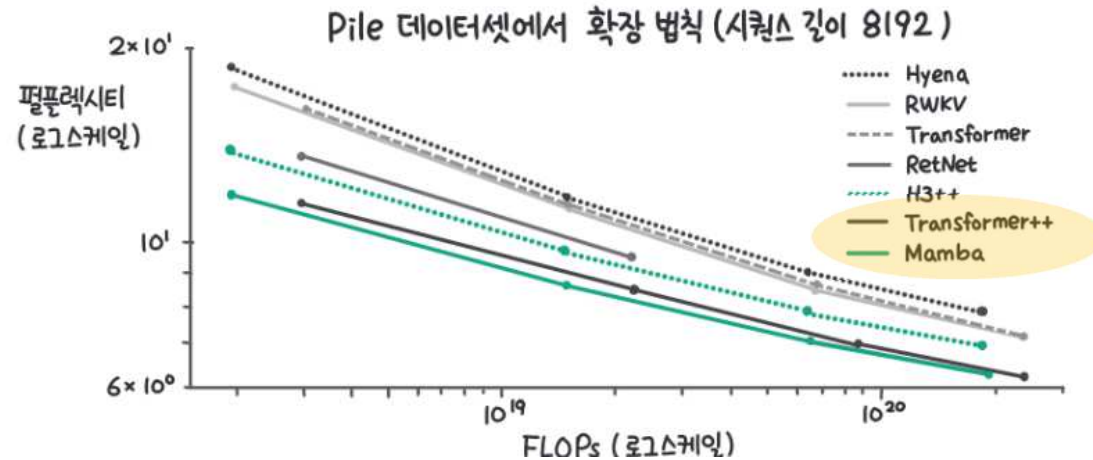
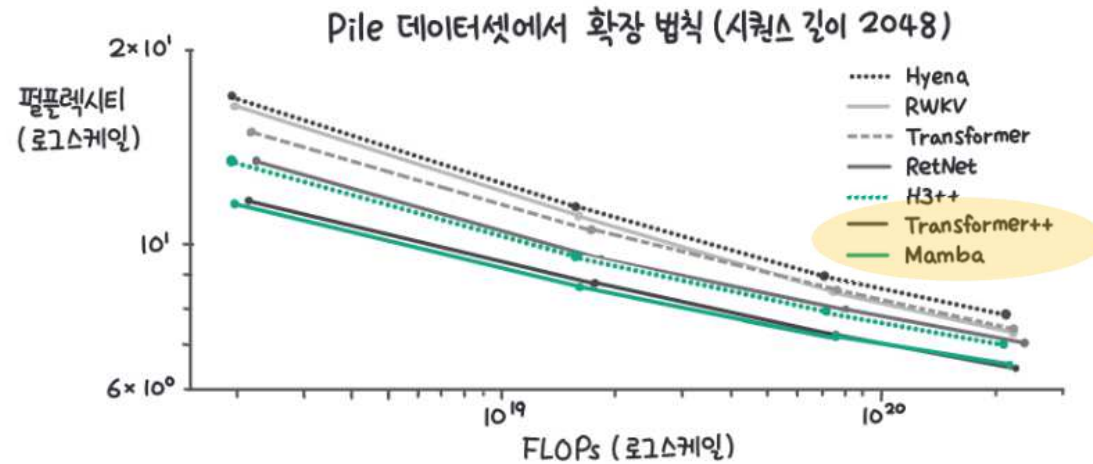
$$h(1) = \mathbf{A}h(0) + \mathbf{B}x(1) = \mathbf{A}\mathbf{B}x(0) + \mathbf{B}x(1)$$

$$h(2) = \mathbf{A}h(1) + \mathbf{B}x(2) = \mathbf{A}^2\mathbf{B}x(0) + \mathbf{A}\mathbf{B}x(1) + \mathbf{B}x(2)$$

맘바의 성능

□ 모델별 성능 비교

- 트랜스포머++와 맘바가 모델 크기와 시퀀스 길이별 대부분 우수한 결과를 보임
so, 연산 부담이 큰 트랜스포머의 강력한 대안으로 주목받고 있음








항목	Transformer (원형)	Transformer++ (최신 개선형)
기본 구조	2017년 Vaswani 등 "Attention is All You Need"	Meta, Google, Mamba 논문 등 최신 기법 통합
Attention	Full attention	FlashAttention, Multi-query attention 등
LayerNorm 위치	Post-LN 또는 Pre-LN	Sandwich-LN, RMSNorm 등 개선
FeedForward	Dense FFN	SwiGLU, Gated Linear Unit (GLU)
Positional Encoding	정적 (sinusoidal, learned)	Rotary, ALiBi, Dynamic Position Bias
병렬성	느림 (seq → seq)	CUDA 최적화 + 커널 퓨전 (FlashAttention2 등)
연산 성능	기준 모델	매우 빠름 (Mamba와 비슷하거나 더 빠름)

아키텍처별 비교

	RNN	트랜스포머	S4	맘바
압축	작은 상태 크기	압축하지 않음(KV 캐시)	작은 상태 크기	선택 메커니즘을 활용한 효율적인 압축
성능	낮음	높음	중간	높음
효율성	높음	낮음	높음	높음
학습 연산량	시퀀스 길이에 비례	시퀀스 길이의 제곱에 비례	시퀀스 길이에 비례	시퀀스 길이에 비례
토큰당 추론 연산량	일정	시퀀스 길이에 비례	일정	일정
병렬화	불가능	가능	가능(컨볼루션 활용)	가능(병렬 스캔 활용)

mamba의 한계는 무엇이며, 극복하기 위해서 어떤 알고리즘이 연구되고 있나

한계점	설명
 시점 간 순차성 (순차 연산)	SSM 기반이지만 여전히 시간 의존 구조 → 완전 병렬화는 어렵고, Transformer만큼의 병렬성은 없음
 정보 흐름 한계	컨볼루션 커널의 범위 제한, 선택 메커니즘(gating)으로 인해 장기 의존성(long-range dependency) 처리에서 Transformer보다 약할 수 있음
 복잡한 구조 학습 어려움	Mamba는 상대적으로 간단한 구조(1D state-space filtering)를 사용하기 때문에, 복잡한 개념/구문 추론에선 Transformer보다 성능이 제한될 수 있음
 대규모 pretraining에 대한 검증 부족	아직 GPT-4, LLaMA 수준의 대규모 언어모델로 학습된 Mamba 계열 모델은 상대적으로 적음
 범용성 부족	이미지, 멀티모달 등 다른 영역에는 아직 확장 사례가 적음 (언어/시 계열 위주)

❑ Mamba++ / Gated Mamba / Structured Mamba

더 깊은 선택 메커니즘 또는 다중 SSM 커널을 적용하여 정보 흐름을 풍부하게 만들

❑ Retentive Network 계열 (RetNet 등)

SSM과 Transformer의 장점을 결합한 모델
순차성은 유지하면서도 부분 병렬화가 가능

❑ Hybrid 모델: Mamba + Attention

SSM 구조에 self-attention 블록을 섞는 방식도 제안
gated-attention + Mamba filter → 장기 의존성과 단기 응답성 모두 확보