Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Paper Review for SnSec Lab Seminar

이채영
(BS AI Engineering @ SMWU. | SNSec. Lab)

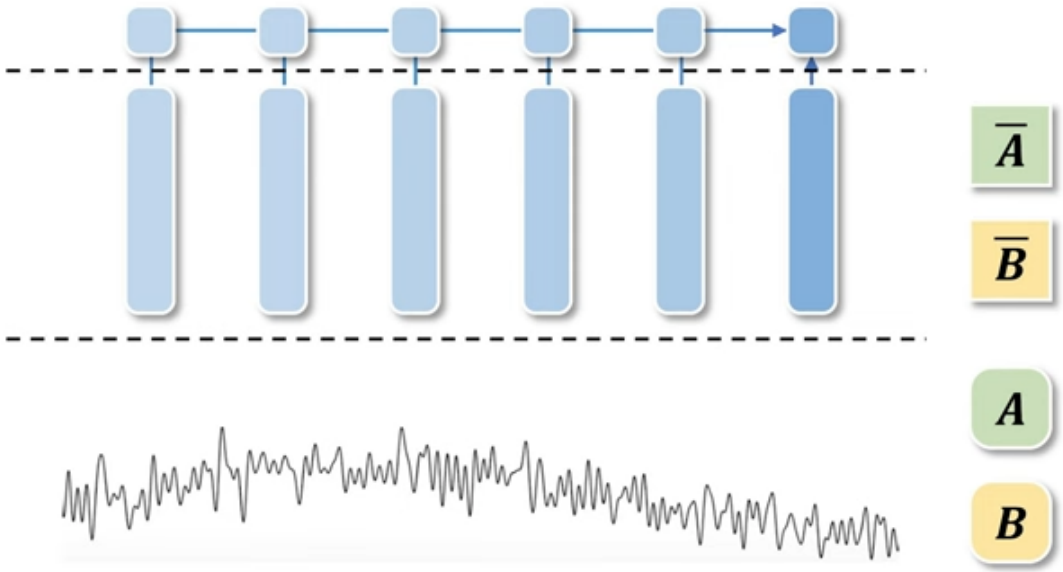
2026년 1월 23일

1. Preliminary: 왜 Mamba인가?

- **Transformer의 한계:** 어텐션(Attention)은 $O(L^2)$ 복잡도를 가지며, 추론 시 토큰 생성이 선형적으로 증가하고 메모리 요구량도 늘어남.
- **RNN의 한계:** 히든 스테이트 업데이트의 병렬화가 불가능하여 훈련 속도가 느리고, 유한한 메모리에 긴 정보를 담기 어려워 성능 한계 존재.
- **Mamba의 지향점:** RNN의 빠른 추론(Fast Inference)과 Transformer의 성능 및 빠른 훈련(Parallel Training) 장점을 모두 취하고자 함.

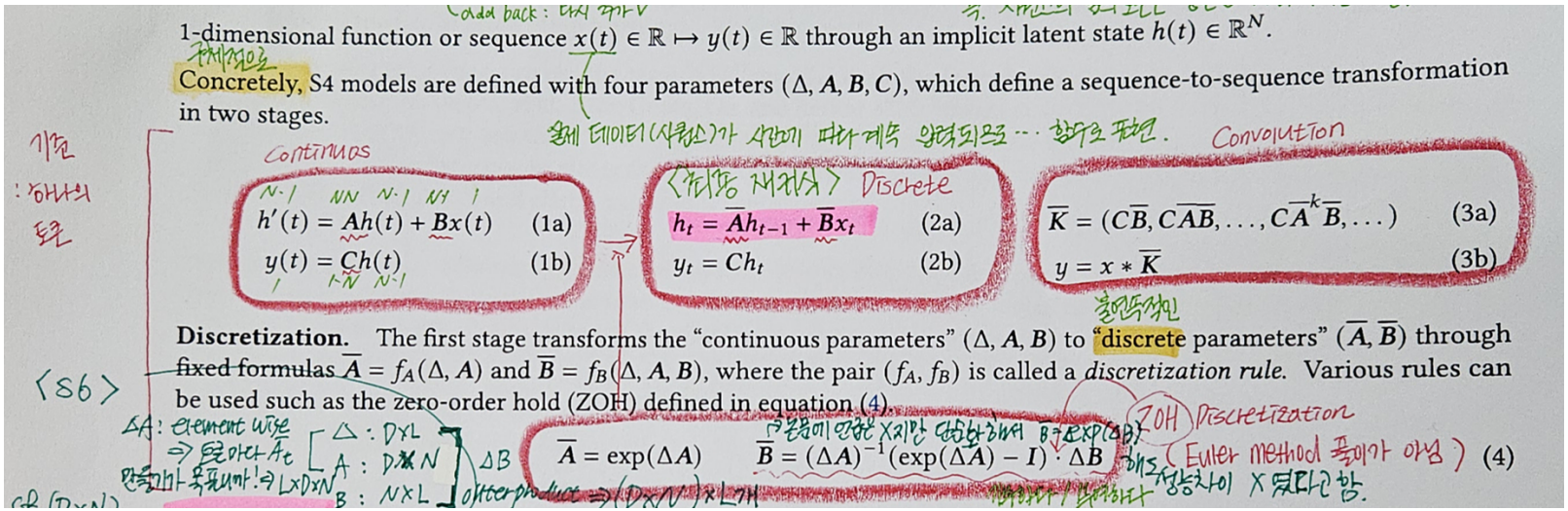
2. State Space Models (SSM) 기초

- **Continuous한 문제 vs Discrete한 문제: 음성, 영상, ... <-> Language Modeling**



2. State Space Models (SSM) 기초

- **시스템 정의:** 입력 $x(t)$를 받아 레이턴트 $h(t)$를 거쳐 출력 $y(t)$를 내는 "연속적" 시간 도메인 시스템.
- **핵심 수식:**
  - $h'(t) = Ah(t) + Bx(t)$ (스테이트 업데이트)
  - $y(t) = Ch(t) + Dx(t)$ (출력 생성. D는 생략하기도.)
- **파라미터 역할:**
  - A: 메모리(스테이트)를 변화시키는 $N \times N$ 매트릭스.
  - B: 입력 스칼라를 N 차원으로 투영하는 벡터.
  - C: 스테이트를 다시 출력 스칼라로 매핑하는 벡터.



1-dimensional function or sequence $x(t) \in \mathbb{R} \mapsto y(t) \in \mathbb{R}$ through an implicit latent state $h(t) \in \mathbb{R}^N$.

Concretely, S4 models are defined with four parameters (Δ, A, B, C), which define a sequence-to-sequence transformation in two stages.

Continuous

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$

Discrete

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t \quad (2a)$$
$$y_t = Ch_t \quad (2b)$$

Convolution

$$\overline{K} = (C\overline{B}, C\overline{AB}, \ldots, C\overline{A}^k\overline{B}, \ldots) \quad (3a)$$
$$y = x * \overline{K} \quad (3b)$$

Discretization. The first stage transforms the "continuous parameters" (Δ, A, B) to "discrete parameters" $(\overline{A}, \overline{B})$ through fixed formulas $\overline{A} = f_A(\Delta, A)$ and $\overline{B} = f_B(\Delta, A, B)$, where the pair (f_A, f_B) is called a *discretization rule*. Various rules can be used such as the zero-order hold (ZOH) defined in equation (4).

$$\overline{A} = \exp(\Delta A) \qquad \overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B \quad (4)$$

3. 이산화 (Discretization)

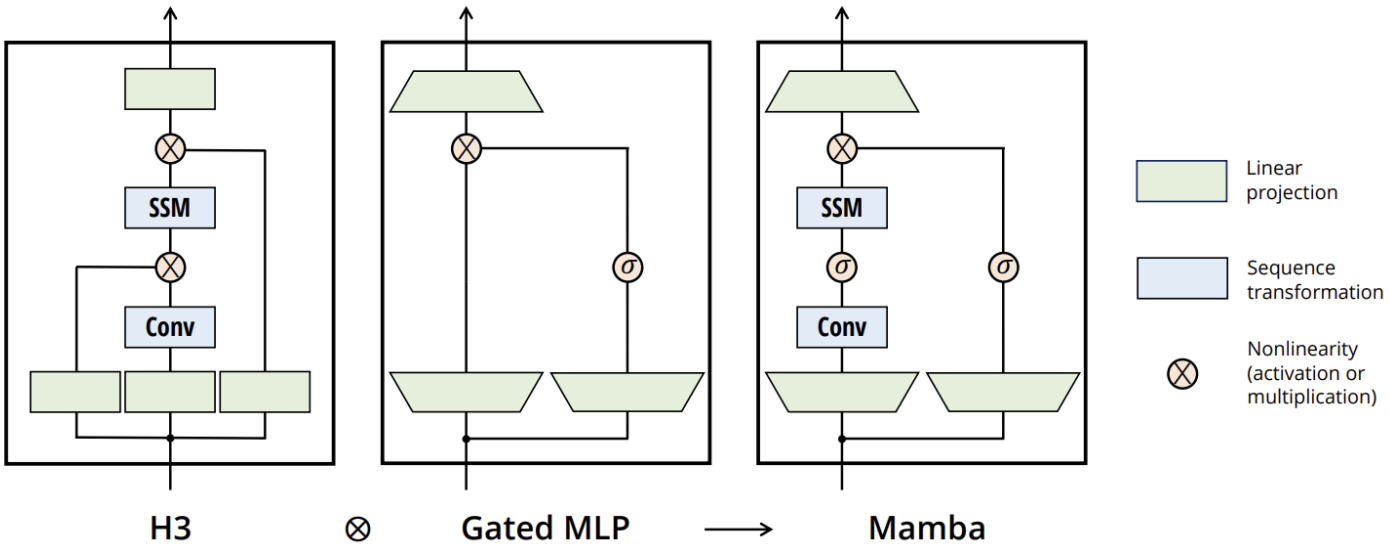
- **방법론:** 연속형 모델을 딥러닝에서 쓰기 위해 이산적 시퀀스 형태로 변환 (Discretization) 필요.
- **Zero-Order Hold (ZOH):**
  - $\bar{A} = \exp(\Delta A)$
  - $\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$
- **결과:** $\bar{A}, \bar{B}$를 통해 $x_t = \bar{A}x_{t-1} + \bar{B}u_t$ 꼴의 RNN recurrence 식 도출 가능.
- **훈련 시:** Linear Time Invariant (LTI) 특성 덕분에 전체 시퀀스를 Convolution 연산으로 병렬 처리 가능.

$$\begin{aligned}\mathbf{x}(t+\Delta) &\cong \Delta(\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)) + \mathbf{x}(t) \\ &= \Delta\mathbf{A}\mathbf{x}(t) + \Delta\mathbf{B}\mathbf{u}(t) + \mathbf{x}(t) \\ &= (\mathbf{I} + \Delta\mathbf{A})\mathbf{x}(t) + \Delta\mathbf{B}\mathbf{u}(t) \\ &= \bar{\mathbf{A}}\mathbf{x}(t) + \bar{\mathbf{B}}\mathbf{u}(t)\end{aligned}$$

$$\begin{aligned}f'(a) &= \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \\ &= \lim_{x \to a} \frac{f(x) - f(a)}{x - a}\end{aligned}$$

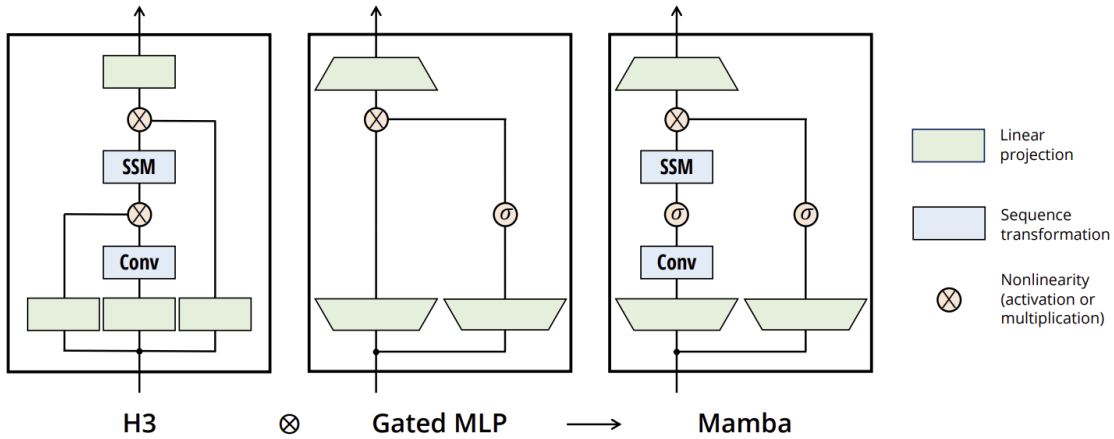
4. Mamba로 가는 길 (Related Works)

- **LSSL:** SSM을 블록처럼 쌓아 성능을 높이려 함.
- **HiPPO:** 이전 시퀀스 히스토리를 잘 기억하기 위해 A 매트릭스를 특정 다항함수(Legendre) 기반으로 초기화하는 기법 제안.
- **S4 (Structured SSM):** A 매트릭스를 대각화(Diagonalization)하여 연산 복잡도를 줄이고 실제 훈련 가능하게 개선.
- **H3:** Transformer의 특정 능력(Inductive Head, Associative Recall)을 위해 Conv1D 레이어를 추가한 구조.



Comparison of Evolution in State Space Models

Model	Mechanism	Complexity	Key Innovation	Limitations
Prior SSM	LTI (Fixed)	$O(N)$	Discretization via Δ.	Unstable; slow serial training.
S4	LTI (Fixed)	$O(N \log N)$	**HiPPO Matrix:** Optimal history memory.	Content-agnostic (no focus).
H3	LTI + Gating	$O(N \log N)$	**Associative Recall:** Mimics Attention.	Still LTI; complex architecture.
Mamba	**LTV (Selective)**	$O(N)$	**Selection Mechanism** & Hardware-aware Scan.	Needs specialized CUDA kernels.



5. Selective SSM의 도입 배경

- **LTI의 문제점:** 기존 SSM은 토큰에 상관없이 똑같은 A, B를 사용하므로, 중요한 토큰만 기억하거나 불필요한 토큰을 잊는 '선택적 복사(Selective Copying)'를 못함.
- **Selective copying:** SSM은 Copying task를 수정하여 토큰의 위치를 변경. 중요한 토큰은 기억하고 불필요한 토큰을 필터링하는 내용 인식 (content-aware) 추론이 필요
- **Induction Heads copying:** 적절한 문맥에서 올바른 출력을 생성하는 데 필요한 내용 인식 추론을 요구. 이전에 발견한 패턴을 추출해서 재현하는 것.

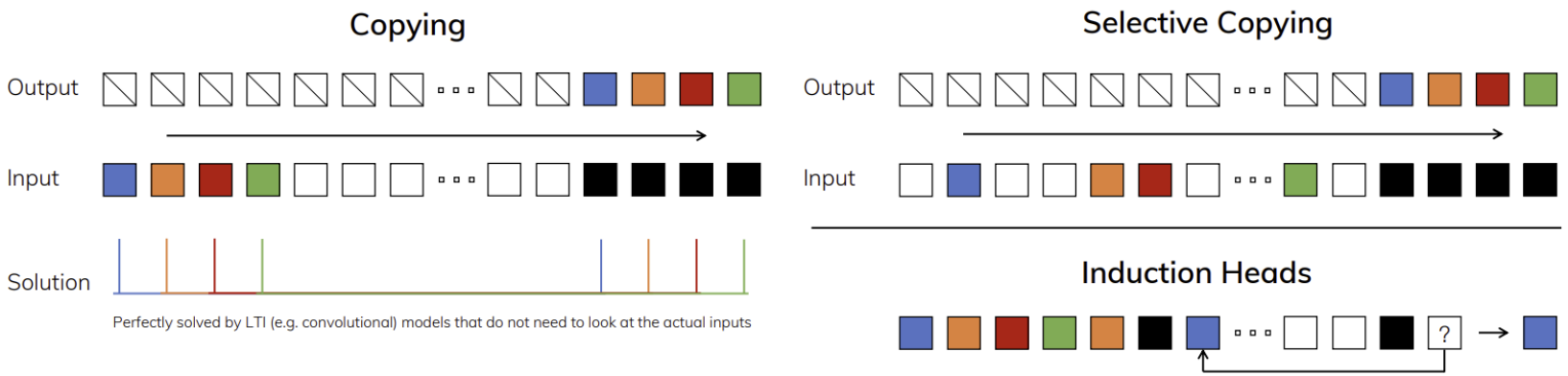


Figure 2: (*Left*) The standard version of the Copying task involves constant spacing between input and output elements and is easily solved by time-invariant models such as linear recurrences and global convolutions. (*Right Top*) The Selective Copying task has random spacing in between inputs and requires time-varying models that can *selectively* remember or ignore inputs depending on their content. (*Right Bottom*) The Induction Heads task is an example of associative recall that requires retrieving an answer based on context, a key ability for LLMs.

5. Selective SSM의 도입 배경

- **H3의 LTI의 문제점 Mamba 논의 시작:** 기존 SSM은 토큰에 상관없이 똑같은 A, B를 사용하므로, 중요한 토큰만 기억하거나 불필요한 토큰을 잊는 '선택적 복사(Selective Copying)'를 못함.
- **해결책 (Selectivity):** 입력 x에 따라 파라미터가 변하는 **데이터 의존적 (Data-dependent)** 구조로 변경.
- **파라미터 함수화:** B_t, C_t, Δ_t를 입력 x_t의 linear projection (직역하면 선형 투영) 결과로 설정.

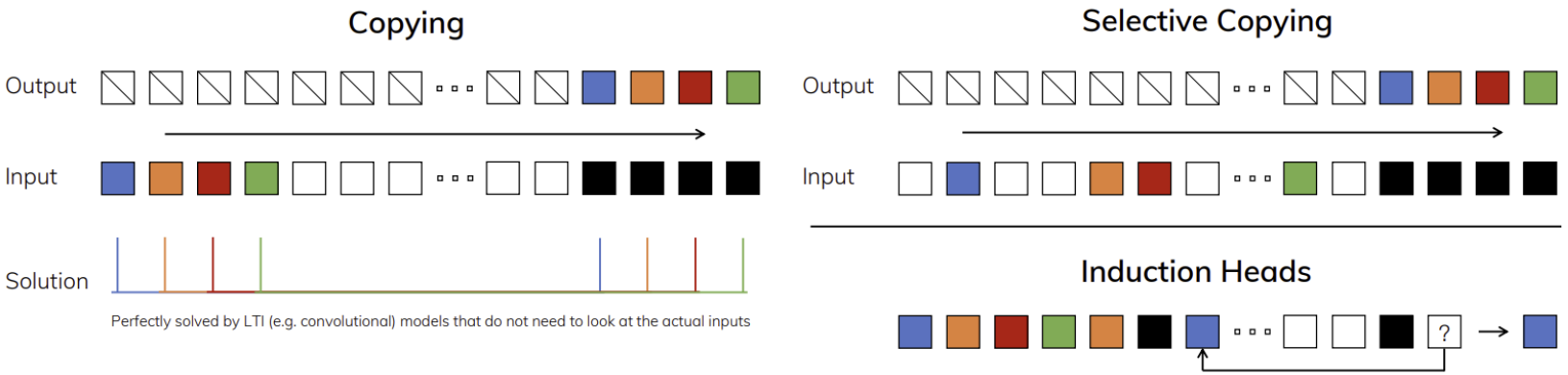


Figure 2: (*Left*) The standard version of the Copying task involves constant spacing between input and output elements and is easily solved by time-invariant models such as linear recurrences and global convolutions. (*Right Top*) The Selective Copying task has random spacing in between inputs and requires time-varying models that can *selectively* remember or ignore inputs depending on their content. (*Right Bottom*) The Induction Heads task is an example of associative recall that requires retrieving an answer based on context, a key ability for LLMs.

6. Selection 여부에 따른 비교 (H3(S4) vs Mamba(S6)

Algorithm 1 SSM (S4)

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

Figure: 실제 구현에서는 엄밀한 ZOH 수식 대신 성능 차이가 적은 간소화된 이산화 수식을 사용하기도 함.

https://github.com/search?q=repo%3Astate-spaces%2Fmamba%20discrettype=code

6. Selection 여부에 따른 비교 (H3(S4) vs Mamba(S6)

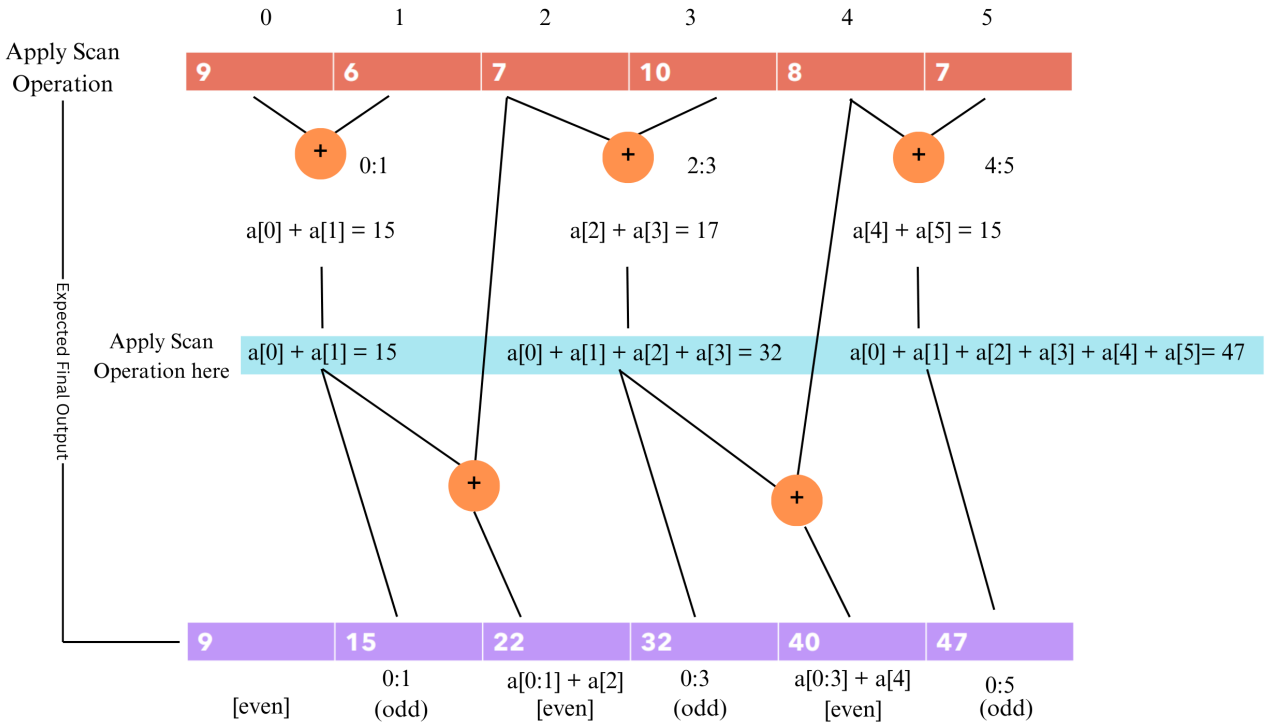
- **Δ의 역할:** 학습된 Δ가 0에 가까우면 이전 스테이트를 그대로 유지하고 현재 입력을 무시, 크면 현재 입력을 더 많이 반영.
- **채널 독립성:** d_{model} 차원의 각 엘리먼트는 독립적인 SSM 시스템으로 취급되어 연산됨.

7. 하드웨어 최적화 (Hardware-Aware Scan) for $O(N)$

- Selective SSM 자체는 본질적으로 Recurrent한 구조이므로 이론적인 연산량은 이미 시퀀스 길이(L 혹은 N)에 비례하는 $O(L)$
- Two challenges와 solutions
- **문제:** sequential nature of reccurrence | large memory usage.

7. 하드웨어 최적화 (Hardware-Aware Scan) for $O(n)$

- Two challenges와 solutions
- **문제 1:** sequential nature of reccurrence).
- **해결 - Parallel (Associative) Scan:** 연산의 결합법칙을 이용해 reccurrent 연산을 로그 타임($O(\log L)$)에 병렬 처리.



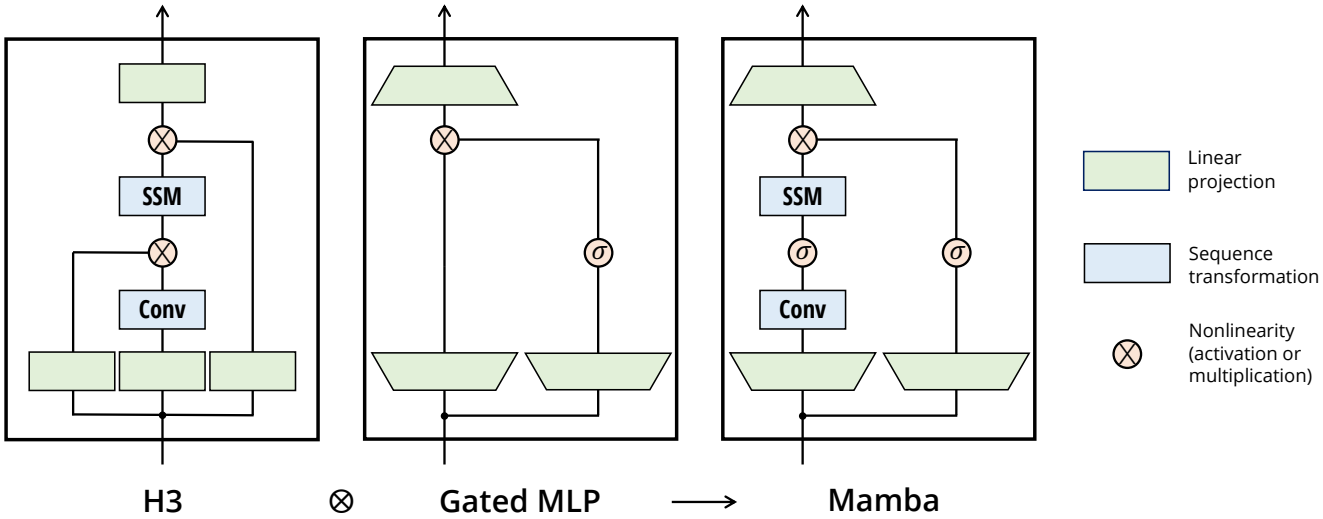
https://blog.premai.io/s4-and-mamba/

7. 하드웨어 최적화 (Hardware-Aware Scan) for $O(n)$

- Two challenges와 solutions
- **문제 2:** large memory usage.
- **해결 - Kernel Fusion:** GPU의 HBM과 SRAM 사이의 텐서 복사(입출력) 비용을 줄이기 위해 이산화와 스캔 연산을 하나의 커널로 합쳐 수행. -> 메모리 병목 해결
- **해결 - Recomputation:** 모든 중간 스테이트를 HBM에 저장하는 대신, 저장하지 않고 버림. 그리고 역전파 단계에서 그 값이 다시 필요해지면, 저장된 입력값으로부터 해당 부분의 연산을 다시 연산하여 값을 얻어냄. 계산량은 늘어나지만, Mamba는 Kernel Fusion을 통해 연산 자체가 매우 빠르기 때문에 memory bottleneck을 해결함으로써 얻는 속도 이득이 재계산으로 인한 손실보다 훨씬 큼. -> 큰 메모리 사용 문제 해결

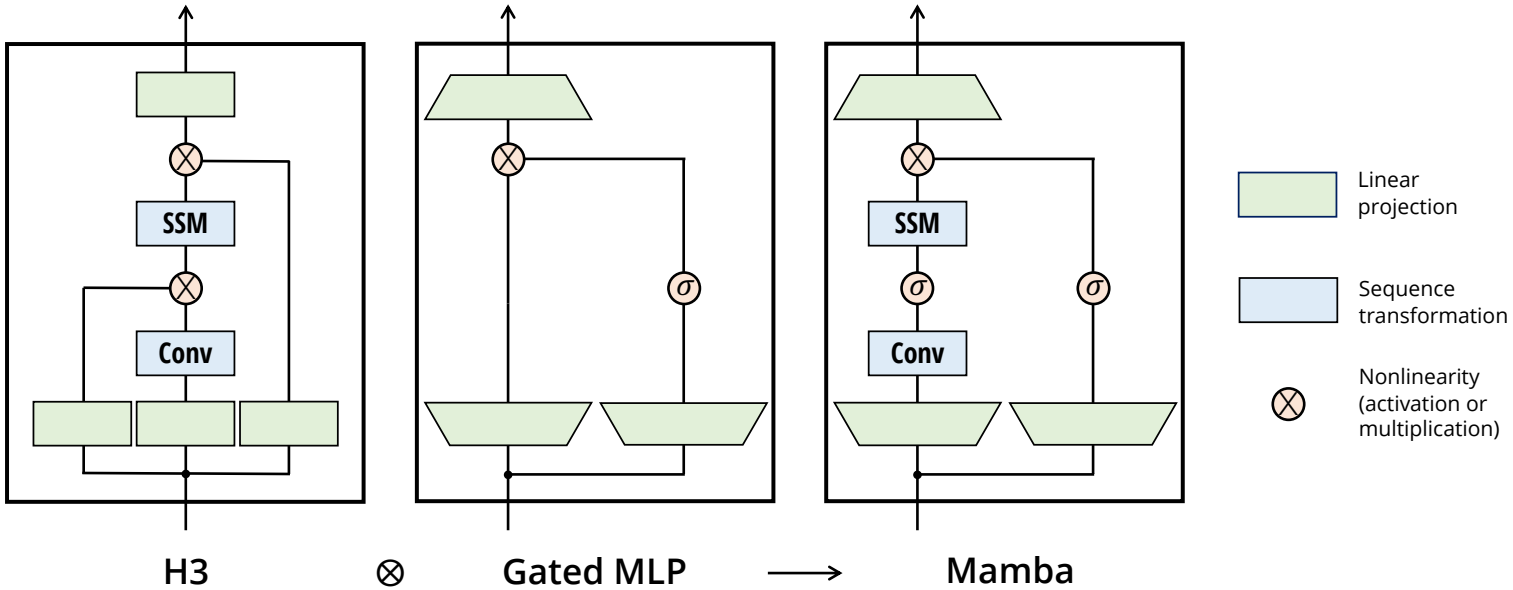
8. Mamba 블록 아키텍처

- **구성:** H3 아키텍처와 Gated MLP 구조를 결합.
- **한 블록 내 흐름:**
  1. 리니어 프로젝션을 통해 d_{model} 차원을 d_{inner}로 확장.
  2. 시퀀스 방향으로 Conv1D 수행 (토큰 간 정보 믹싱).
  3. Selective SSM 적용.
  4. 게이팅(Gating) 연산 후 다시 d_{model}로 축소.
- **블록 간 연결:** 블록 사이에 standard normalization이나 residual connections 적용.



8. Mamba 블록 아키텍처

- **구성:** H3 아키텍처와 Gated MLP 구조를 결합.
- **블록 간 연결:** 블록 사이에 standard normalization이나 residual connections 적용.



8. Mamba 블록 아키텍처

- **residual connections:** ResNet에서 처음 도입된 이후, 현재는 자연어 처리의 Transformer에도 적용. skip connection이라고 부르기도 함. 어떤 복잡한 함수 $H(x)$를 직접 학습하는 것보다, 그 차이인 $F(x) = H(x) - x$를 학습하는 것이 훨씬 쉽다는 컨셉.

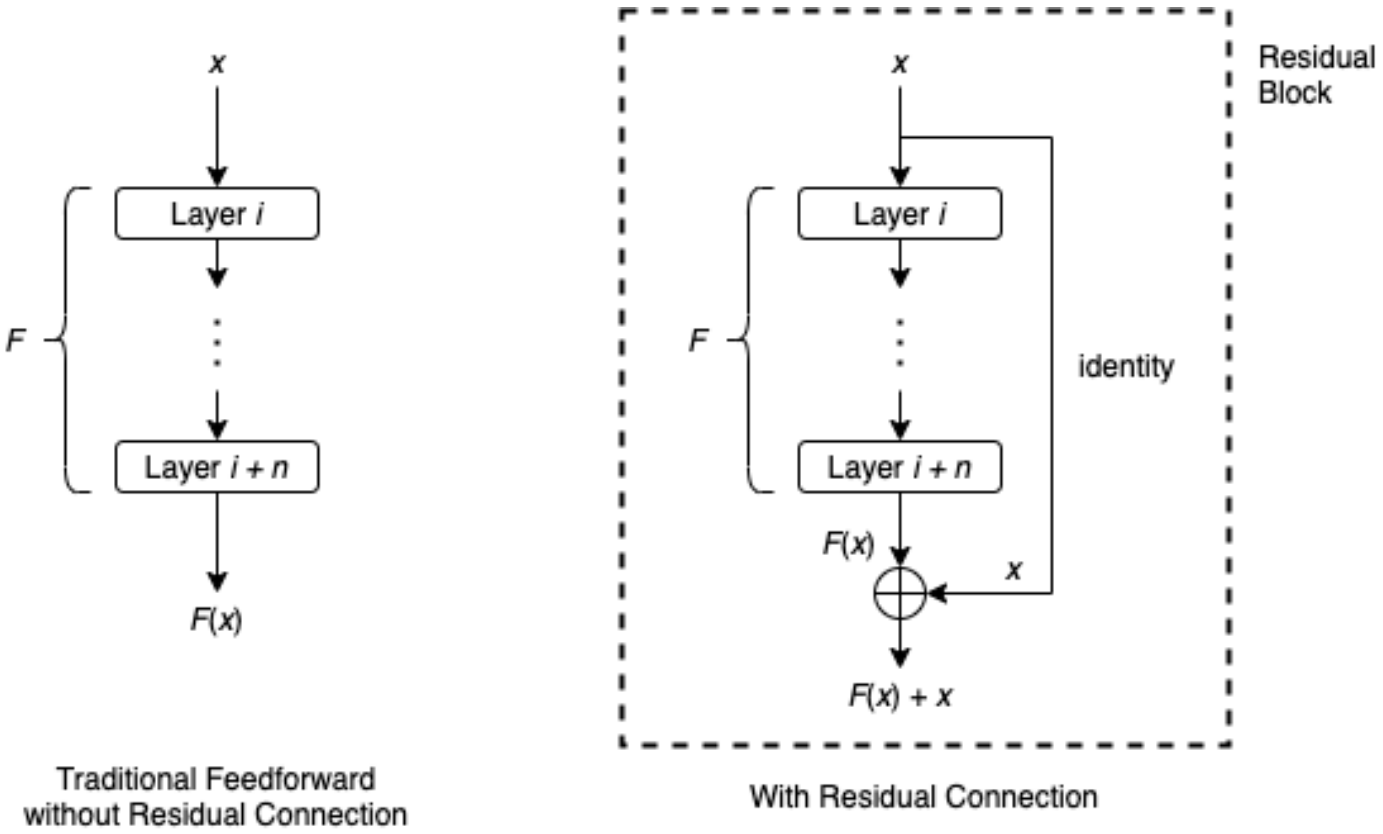
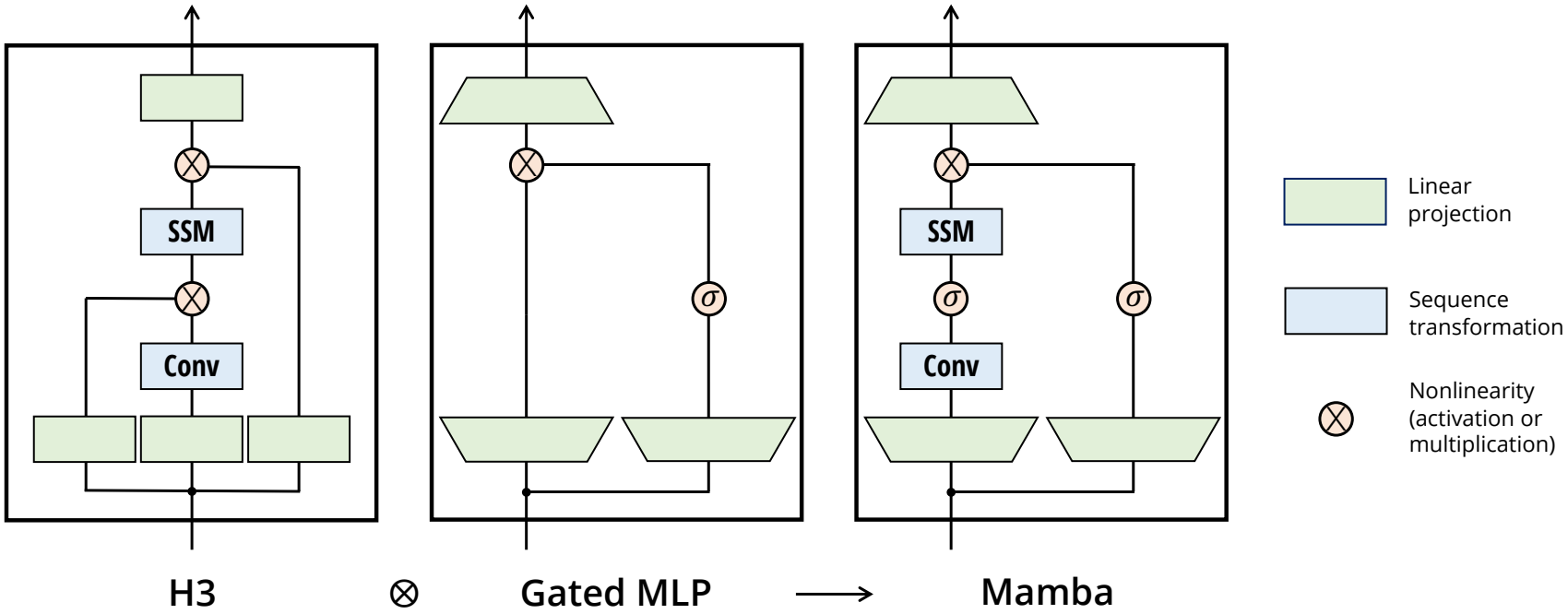


Figure: Enter Caption

8. Mamba 블록 아키텍처

- **한 블록 내 흐름:**
  1. 리니어 프로젝션을 통해 d_{model} 차원을 d_{inner}로 확장.
  2. 시퀀스 방향으로 Conv1D 수행 (토큰 간 정보 믹싱).
  3. Selective SSM 적용.
  4. 게이팅(Gating) 연산 후 다시 d_{model}로 축소.



9. 실험 결과: 합성 태스크 및 스케일링

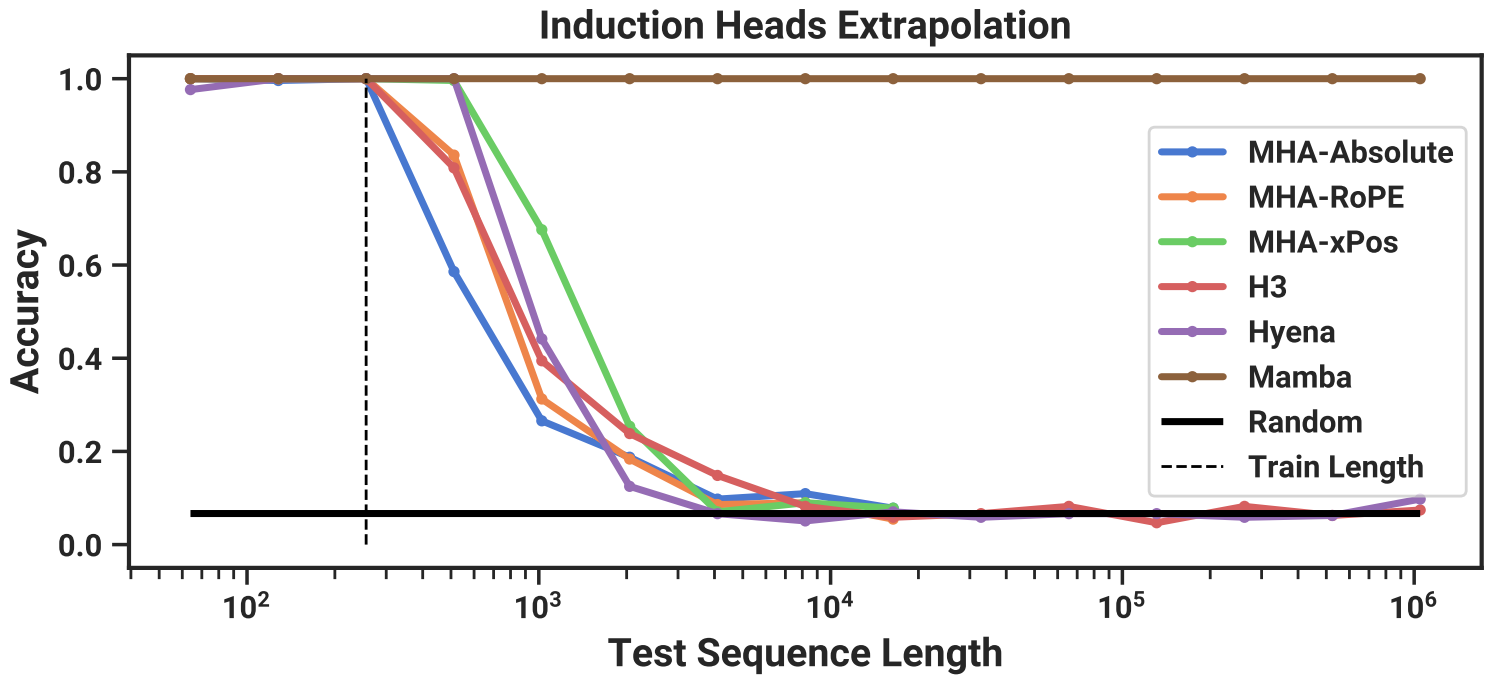
- **Synthetic Tasks:** "Selective Copying" 및 Induction Heads에서 기존 SSM 보다 압도적 성능을 보이며 Transformer 급의 능력 증명.

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.

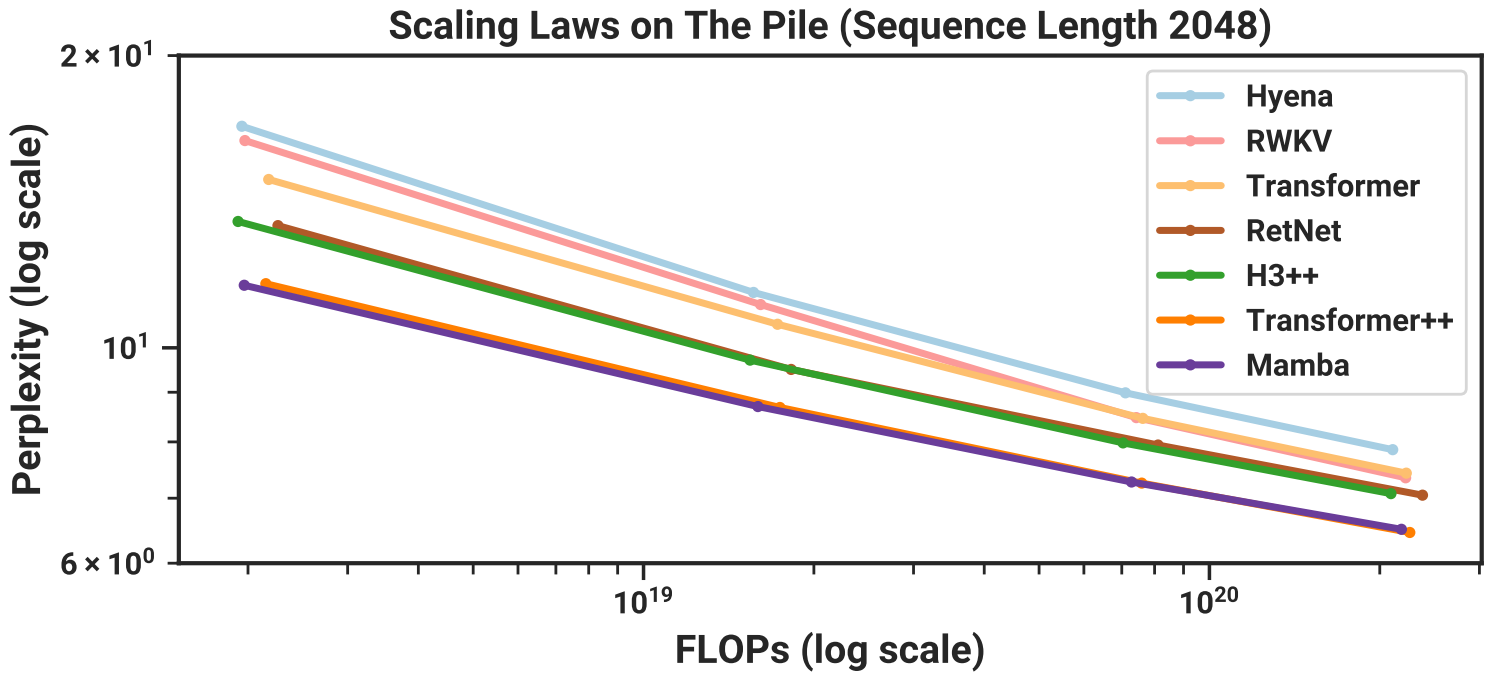
9. 실험 결과: 합성 태스크 및 스케일링

- **Synthetic Tasks:** Selective Copying 및 "Induction Heads"에서 기존 SSM 보다 압도적 성능을 보이며 Transformer 급의 능력 증명.
- **Extrapolation:** 훈련 길이보다 긴 시퀀스에 대해서도 안정적인 성능 유지.



9. 실험 결과: 합성 태스크 및 스케일링

- **Scaling Laws:** 파라미터가 커짐에 따라 성능이 안정적으로 향상되어 Transformer++ 와 비견되는 양상을 보임.



10. 실험 결과: 속도 및 다운스트림

- **다운스트림 테스크:** 동일 파라미터 규모의 모델들과 성능 비교.

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

10. 실험 결과: 속도 및 다운스트림

- **훈련 속도:** 긴 시퀀스(2k 이상)에서 Flash Attention 2보다 빠른 속도 달성.

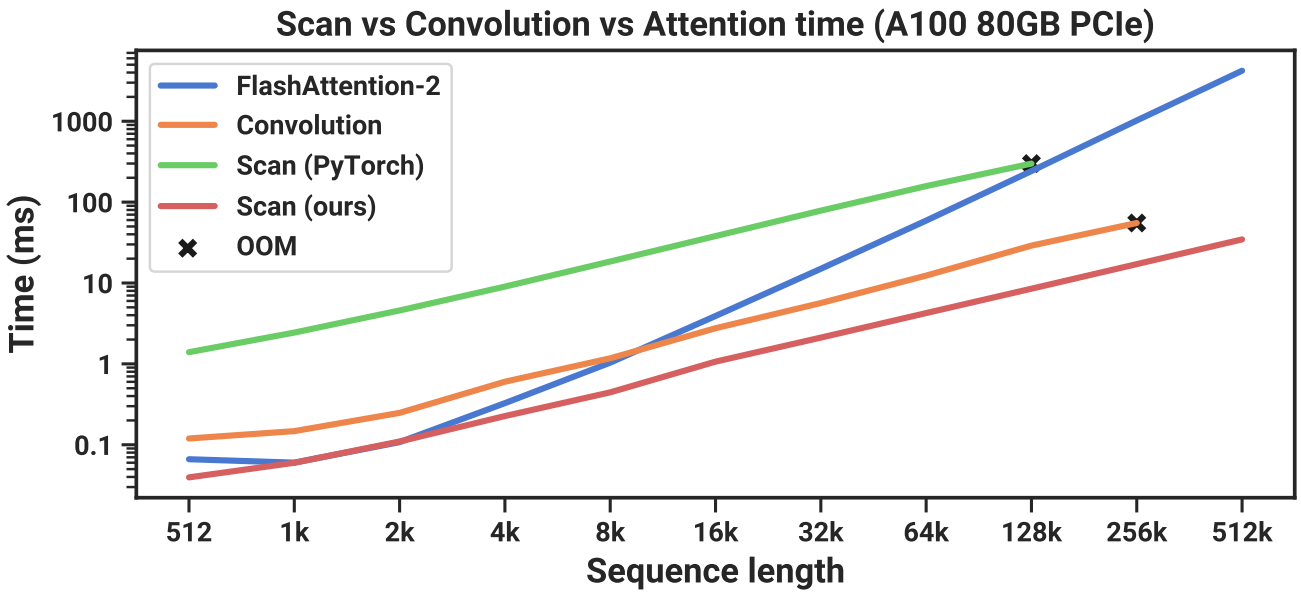


Figure: 시퀀스 길이(x)가 증가함에 따라 훈련 속도(y)의 변화를 관찰함.

10. 실험 결과: 속도 및 다운스트림

- **추론 속도:** 6.9B 모델의 추론 속도가 Transformer 1.3B 모델과 유사할 정도로 월등함.

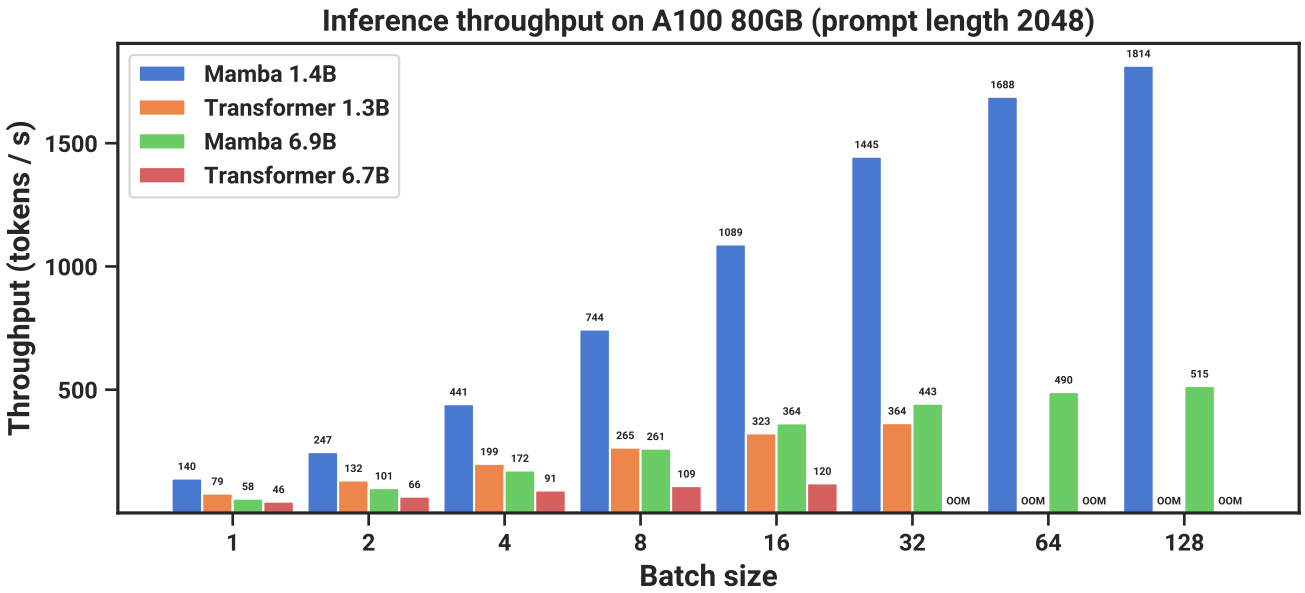


Figure: Mamba와 Transformer의 파라미터수를 달리한 모델(각 2개)을 비교한 그래프. 가로축: 배치 사이즈, 세로축: 추론속도

11. 총정리 및 개인적 생각

- **결론:** 성능, 훈련 속도, 추론 속도를 모두 잡은 모델.
- **한계 및 과제:**
  - 미분과 *Delta*와 아래첨자$(t, t-1)$와 관련한 대응(correspondence) 관계 의문. Mamba 저자들의 논리 전개인지, 기존에 증명된 사항인지는 확인되지 않음.
  - E라는 확장 계수의 필요성이 서술되지 않음. 또한 E값 결정에 Transformer의 파라미터 수를 근거로 한 것은 단순히 추후 비교를 하기 위해서인지 궁금함.
- **의의:** general하게, 보통 좋은 성능을 내는 모델을 설계하기 위해서는 모델이 매우 크거나(=좋은HW), 데이터가 매우 많으면 된다고 생각할 수 있다. 이를 벗어나 $O(L^2)$의 복잡한(큰) 모델 설계를 벗어났다는 점이 인상 깊음. 단순 Application이 아닌 Hardware-aware한 모델 연구라는 점이 매우 흥미로움.