

Linear-Time Sequence Modeling with Selective State Spaces

M. Soleymani

Sharif University of Technology

Spring 2026

Most Visualizations are from: M. Grootendorst, **A Visual Guide to Mamba and State Space Models**

Outline

- Why long text generation creates an inference bottleneck.
- Why RNNs and classical SSMs are natural alternatives.
- The limitation of LTI SSMs: fixed parameters compress all tokens similarly.
- Mamba/S6: selective, input-dependent state-space dynamics.
- Hardware-aware selective scan: keeping the model parallel and efficient.
- Synthetic tasks, results, limitations, and final takeaways.

Motivation: text generation is sequential

- Autoregressive models generate one token at a time.
- Long responses require repeatedly using an expanding context.
- For long generations, speed and memory become central modeling constraints.

Transformers: quality comes from dense interaction

- Self-attention gives each token direct access to previous tokens.
- This makes retrieval and content-based reasoning easy.
- But dense token-token interaction makes long contexts expensive.

Attention : $O(L^2)$ training, $O(L)$ new work/token

Transformer Generation Cost

Transformers for Text Generation

- Transformers are powerful in part because of attention
 - no compression
 - path length of 1 at all times
- Slow inference for long lengths $O(L^2)$

Curse with Inference!

When generating the next token, we need to re-calculate the attention for the *entire sequence*, even if we already generated some tokens.



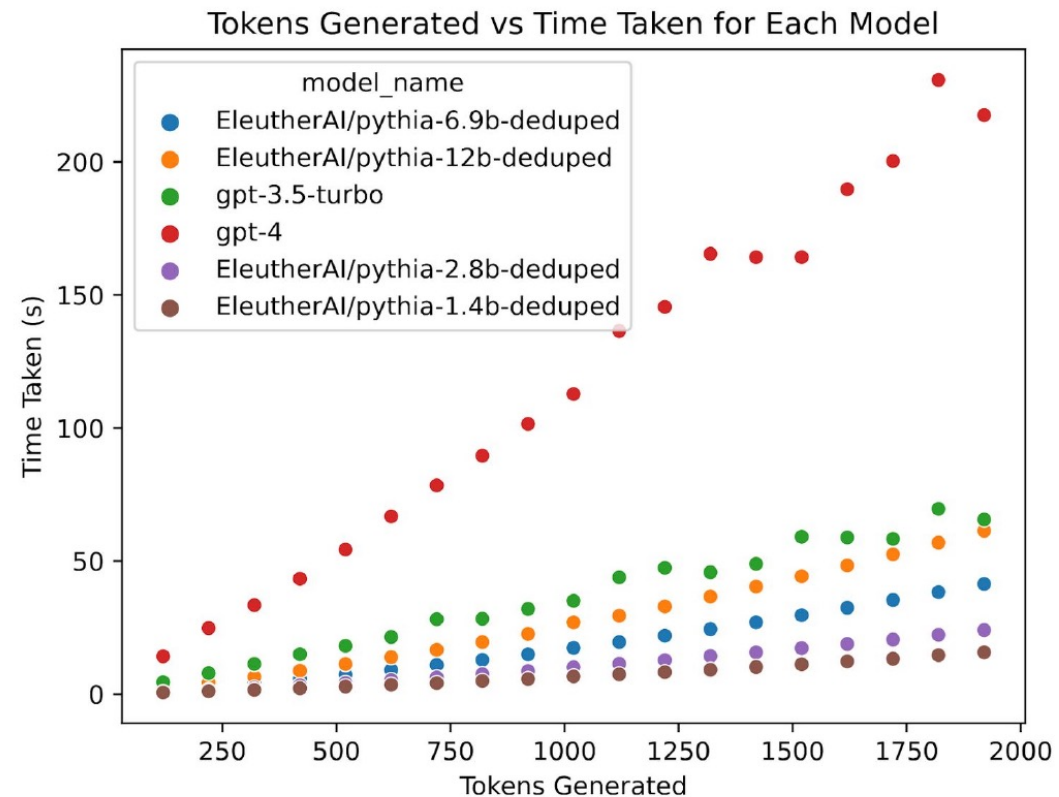
Long Generation Is the Bottleneck

- The concern is not only training cost; long inference and prefill are also expensive.
- Practical throughput depends on both FLOPs and memory movement.
- Mamba targets linear-time sequence mixing with recurrent inference.

Long-Generation Timing Bottleneck

Transformers for Text Generation

Mainly concerned with
long text lengths



Bounded Regret: What Will GPT-2030 Look Like?

Speedups help, but Mamba changes the architecture

- Sparse and local attention reduce the number of interactions.
- FlashAttention-style algorithms reduce memory IO and improve practical speed.
- Mamba asks a different question: can a recurrent state replace explicit attention for long sequences?

Core question

How can we reduce long-sequence inference cost without giving up content-dependent reasoning?

Mamba answer: selective state-space recurrence + hardware-aware scan

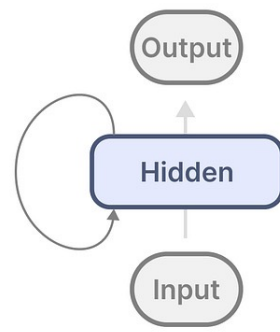
RNNs: compress context into a hidden state

- An RNN stores information in a hidden state rather than in all pairwise token interactions.
- Inference can be constant-state and online.
- Classical RNNs are difficult to train in parallel and often struggle with very long dependencies.

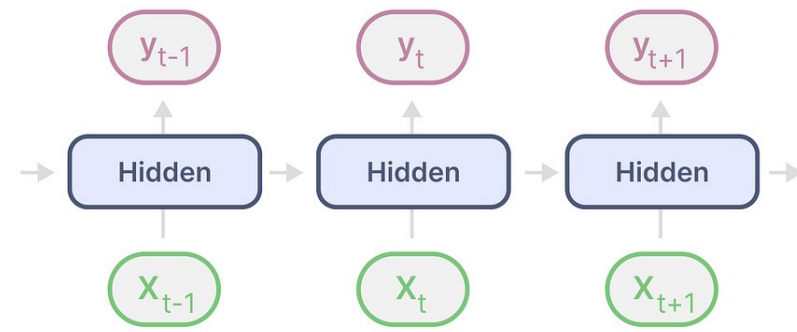
$$h_t = f_{\theta}(h_{t-1}, x_t), \quad y_t = g_{\theta}(h_t)$$

RNN Hidden-State Compression

- Compress information into a hidden state
- Questionable performance on very long sequences
 - Can't train in parallel



RNN

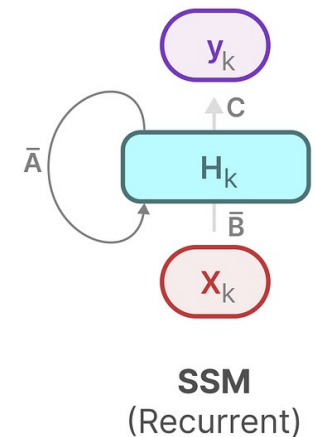


RNN
(Unfolded)

State Space Models: Recurrence with Linear Dynamics

- SSMs also compress the past into a latent state.
- The update is linear in the state, with matrices controlling memory, input injection, and readout.
- This structure makes analysis and efficient implementation possible.

$$\begin{aligned}h_{t+1} &= \bar{A}h_t + \bar{B}x_t \\ y_t &= Ch_t + Dx_t\end{aligned}$$



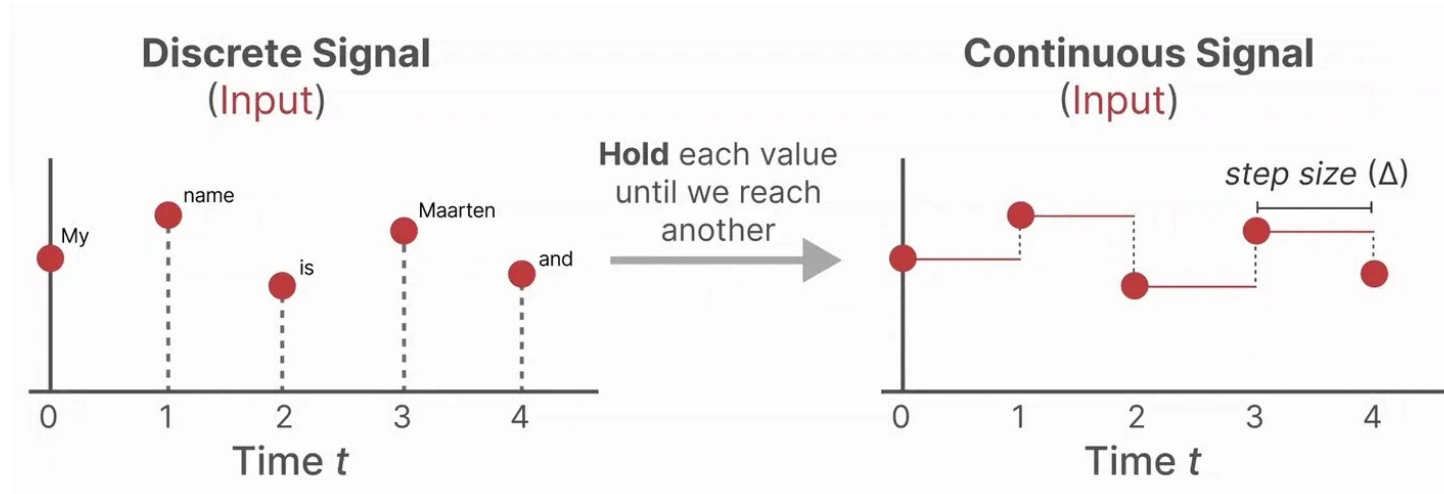
Continuous State Space Models

$$\dot{h}(t) = Ah(t) + Bx(t)$$

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

From a Continuous to a Discrete Signal

Zero-Order Hold (ZOH)



Continuous-Time to Discrete-Time

- Discretization uses a step size Δ to convert continuous dynamics into sequence updates.
- Convert discrete data to continuous using Δ

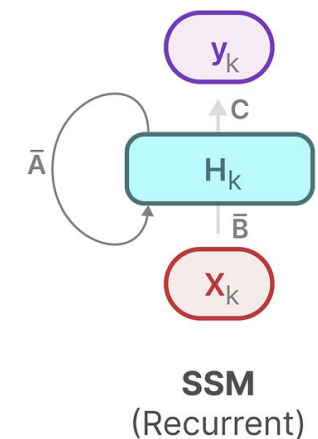
$$\dot{h}(t) = Ah(t) + Bx(t)$$

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

$$h_{t+1} = \bar{A}h_t + \bar{B}x_t$$

$$y_t = Ch_t + Dx_t$$

$$\bar{A} = e^{\Delta A}, \quad \bar{B} = \left(\int_0^{\Delta} e^{\tau A} d\tau \right) B$$



$$\bar{B} = (\Delta A)^{-1}(e^{\Delta A} - I)(\Delta B)$$

Discrete-Time and Convolution View: S4

- If A , B , C are constant across time, the layer is LTI and can be computed as a convolution.

Parallelizable via convolution

$$\begin{aligned}h_{t+1} &= \bar{A}h_t + \bar{B}x_t \\ y_t &= Ch_t + Dx_t\end{aligned}$$

$$h_{t+1} = \bar{A}^2 h_{t-1} + \bar{A}\bar{B}x_{t-1} + \bar{B}x_t$$

$$h_{t+1} = \bar{A}^3 h_{t-2} + \bar{A}^2 \bar{B}x_{t-2} + \bar{A}\bar{B}x_{t-1} + \bar{B}x_t$$

$$h_t = \sum_{k=0}^t \bar{A}^k \bar{B}x_{t-k}$$

$$y_t = \sum_{k=0}^t C \bar{A}^k \bar{B}x_{t-k}$$

Discrete-Time and Convolution View: S4

- If A, B, C are constant across time, the layer is LTI and can be computed as a convolution.

Parallelizable via convolution

$$\begin{aligned}h_{t+1} &= \bar{A}h_t + \bar{B}x_t \\ y_t &= Ch_t + Dx_t\end{aligned}$$

$$h_{t+1} = \bar{A}^2 h_{t-1} + \bar{A}\bar{B}x_{t-1} + \bar{B}x_t$$

$$h_{t+1} = \bar{A}^3 h_{t-2} + \bar{A}^2 \bar{B}x_{t-2} + \bar{A}\bar{B}x_{t-1} + \bar{B}x_t$$

$$h_t = \sum_{k=0}^t \bar{A}^k \bar{B}x_{t-k}$$

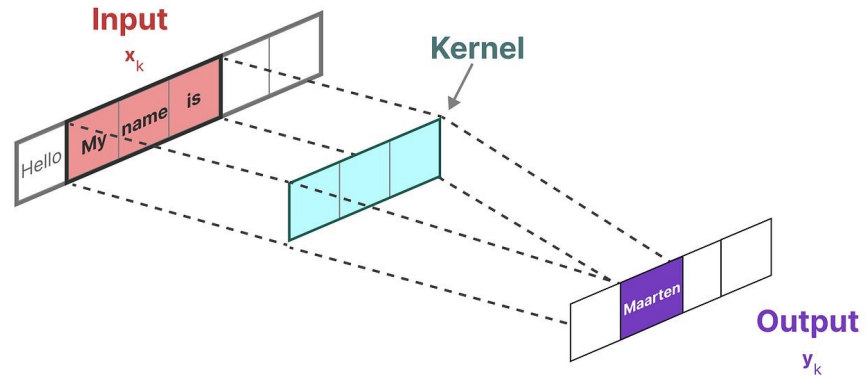
$$y_t = \sum_{k=0}^t C\bar{A}^k \bar{B}x_{t-k}$$

$$K_k = C\bar{A}^k \bar{B}$$

$$y_t = \sum_{k=0}^t K_k x_{t-k}$$

$$\begin{aligned}\bar{K} &= (C\bar{B}, C\bar{A}\bar{B}, \dots, C\bar{A}^k \bar{B}, \dots) \\ y &= x\bar{K}\end{aligned}$$

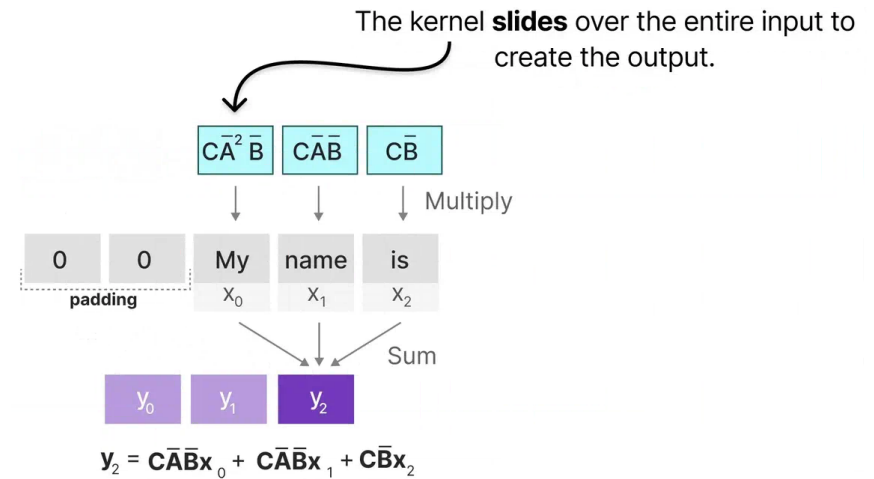
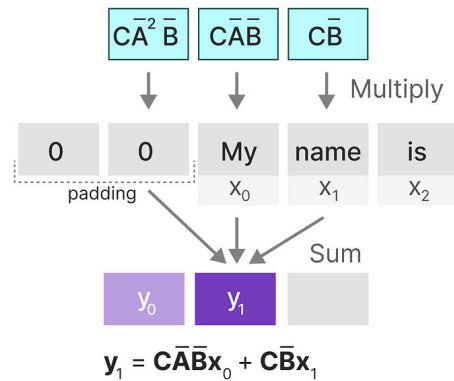
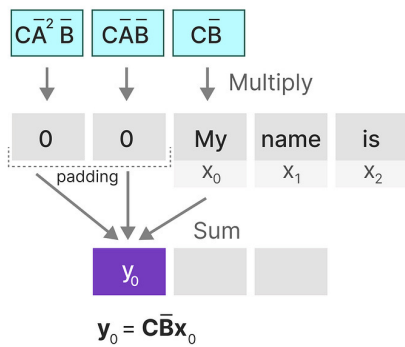
Discrete-Time and Convolution View



Kernel

Input (x_k)

Output (y_k)



Problem: Fixed Dynamics Treat All Tokens Similarly

- LTI SSMs compress all previous tokens into one hidden state.
- The same transition and readout parameters are used at every token.
- This helps parallel convolution, but limits content-dependent behavior.
- Language often requires deciding which tokens to keep, ignore, or compare.

$\bar{A}, \bar{B}, C$ fixed across tokens $\Rightarrow$ time invariant

Mamba restores content sensitivity by making parts of the SSM input-dependent.

Key Idea: Selective State Space Models

- Mamba paper: the fundamental problem is compressing context into a smaller state.
- Selectivity means choosing what information should enter, persist in, or leave the hidden state.
- This is why selective SSMs are sometimes described as attention-like information filtering.

Selection = content-dependent control over the recurrent memory.

Mamba: Block Overview

- A Mamba block contains local convolution, a selective SSM core, and multiplicative gating.
- At the core is S6: an SSM whose parameters vary with the input token.
- The architecture is attention-free, but still content-sensitive.

Selective State Space Model in Mamba

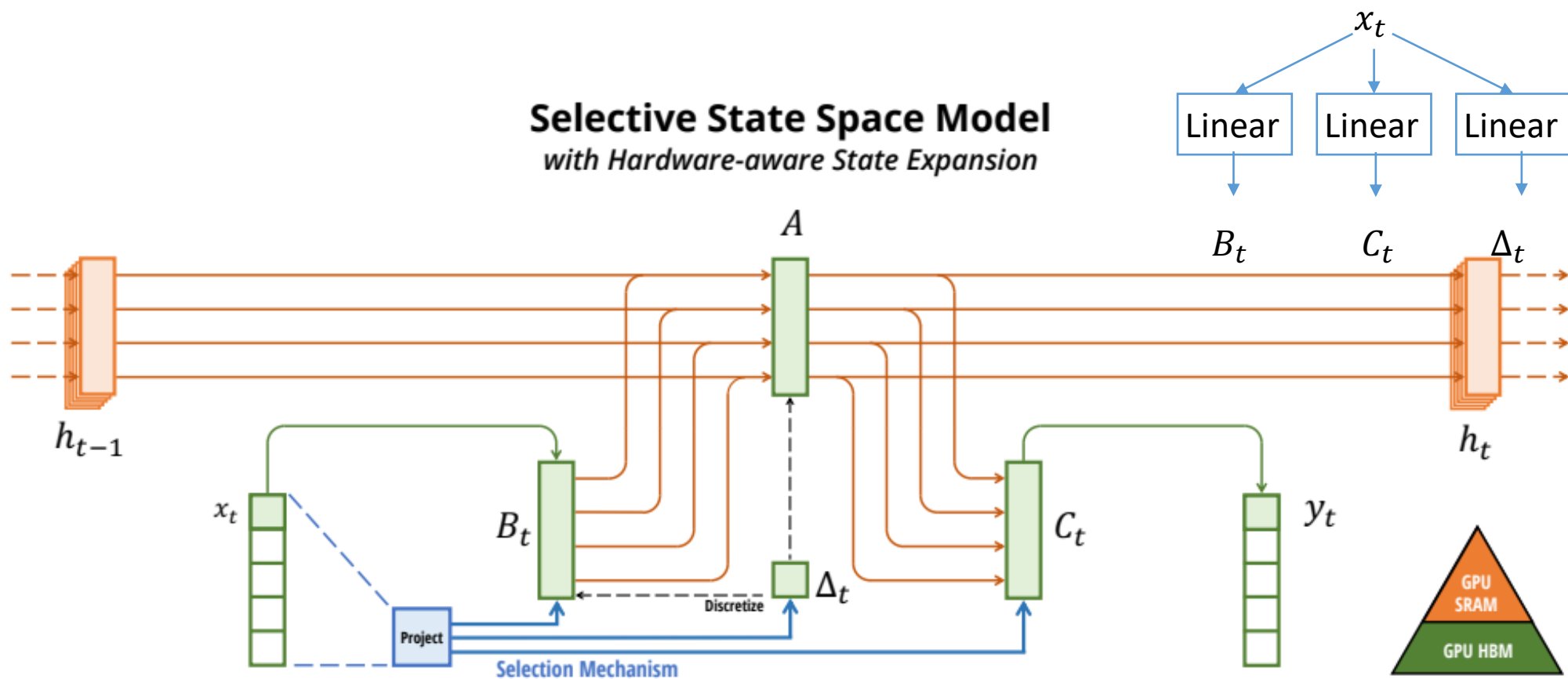


Figure 1: **(Overview.)** Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN , times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the (Δ, A, B, C) parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.

Selective Parameters: Δ , B , C Vary per Token

$$\Delta_t = s_\Delta(x_t), \quad B_t = s_B(x_t), \quad C_t = s_C(x_t)$$

- Δ controls the discretization/time step for each token.

$$h_t = \bar{A}_t h_{t-1} + \bar{B}_t x_t, \quad y_t = C_t h_t$$

- B controls how input enters the state.

$$\bar{A}_t = e^{\Delta_t A}, \quad \bar{B}_t = (\Delta_t A)^{-1} (e^{\Delta_t A} - I) \Delta_t B_t$$

- C controls how the current state is read into the output.

Selective Focus and Ignore Mechanism

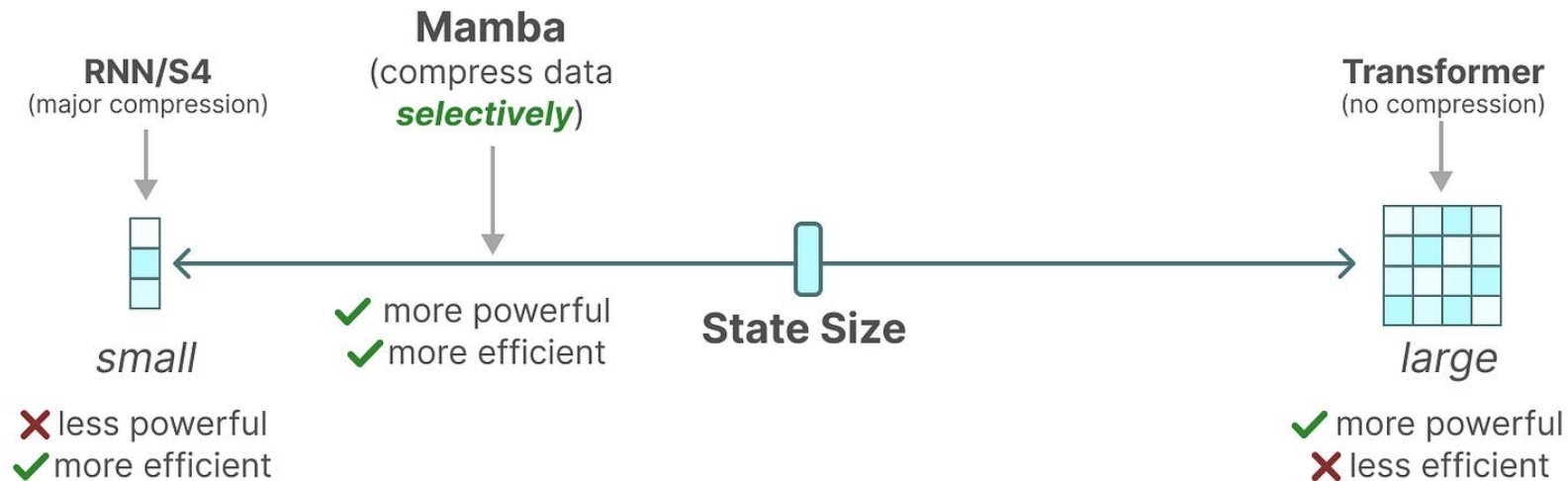
- For each input token, selective parameters determine whether it should affect the hidden state.
- Irrelevant tokens can be suppressed; relevant tokens can be preserved for later prediction.
- This gives an attention-like capability while keeping recurrent inference.

Selective Focus and Ignore Mechanism

- They *selectively* choose what to keep in the hidden state and what to ignore since they are now dependent on the input.
 - A smaller Δ results in ignoring current inputs and instead using the previous context more
 - A larger *step size* Δ focuses on the inputs more than the context:

Selectively Retain Information

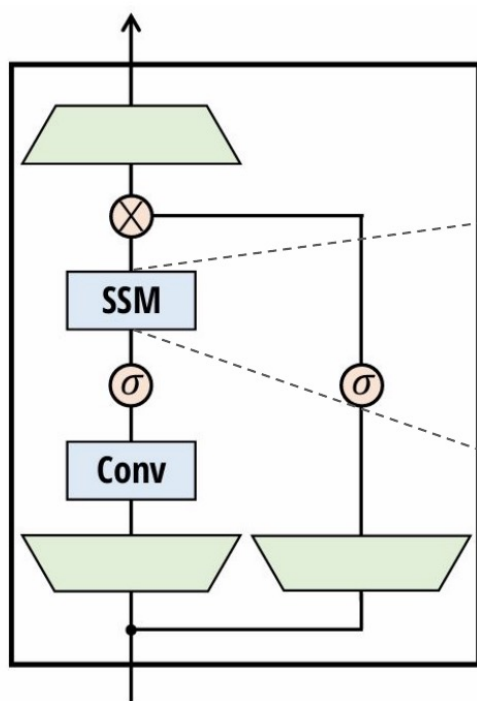
- SSM creates a small state that is quite efficient as it compresses the entire history.
- Compared to a Transformer model, it is much less powerful.
- Mamba: A small state that is as powerful as the state of a Transformer:



- It does so by compressing data selectively into the state.
 - To selectively compress information, the parameters are dependent on the input.

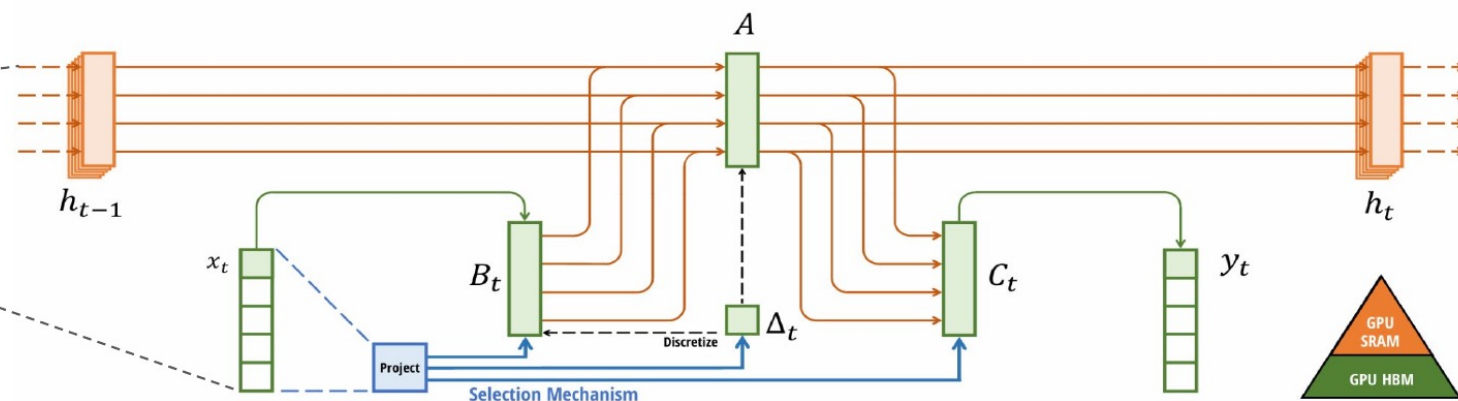
Mamba Block

Mamba

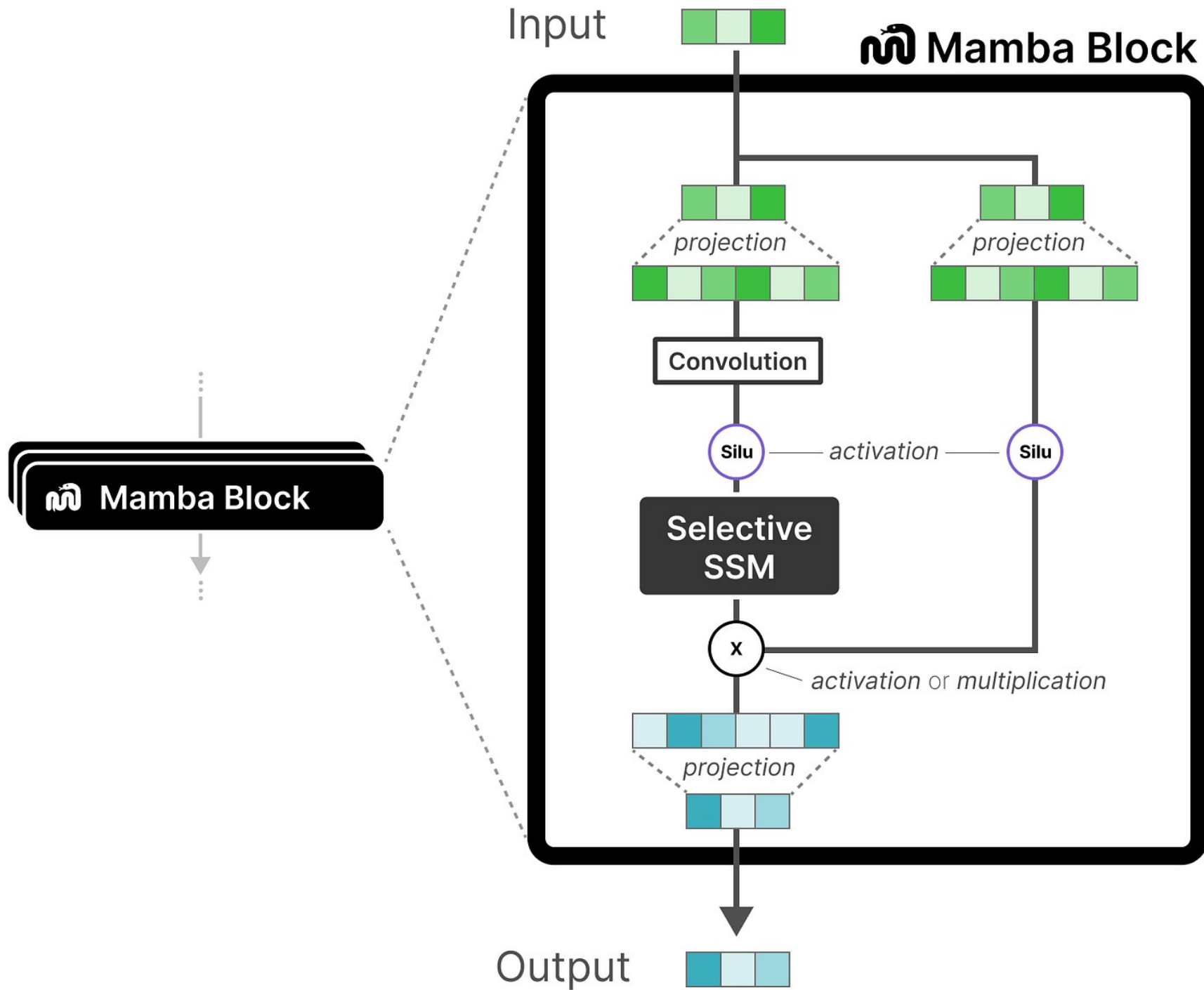


Mamba has both:

1. a local temporal convolution
2. a recurrent/state-space temporal mechanism



Mamba block and at its core Selective SSM, or S6



S4 versus S6 Algorithmic Difference

Algorithm 1 SSM (S4)

Input: $x : (B, L, D)$

Output: $y : (B, L, D)$

1: $A : (D, N) \leftarrow \text{Parameter}$

▷ Represents structured $N \times N$ matrix

2: $B : (D, N) \leftarrow \text{Parameter}$

3: $C : (D, N) \leftarrow \text{Parameter}$

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

5: $\overline{A}, \overline{B} : (D, N) \leftarrow \text{discretize}(\Delta, A, B)$

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

▷ Time-invariant: recurrence or convolution

7: **return** y

Algorithm 2 SSM + Selection (S6)

Input: $x : (B, L, D)$

Output: $y : (B, L, D)$

1: $A : (D, N) \leftarrow \text{Parameter}$

▷ Represents structured $N \times N$ matrix

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

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

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

5: $\overline{A}, \overline{B} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, A, B)$

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

▷ **Time-varying:** recurrence (*scan*) only

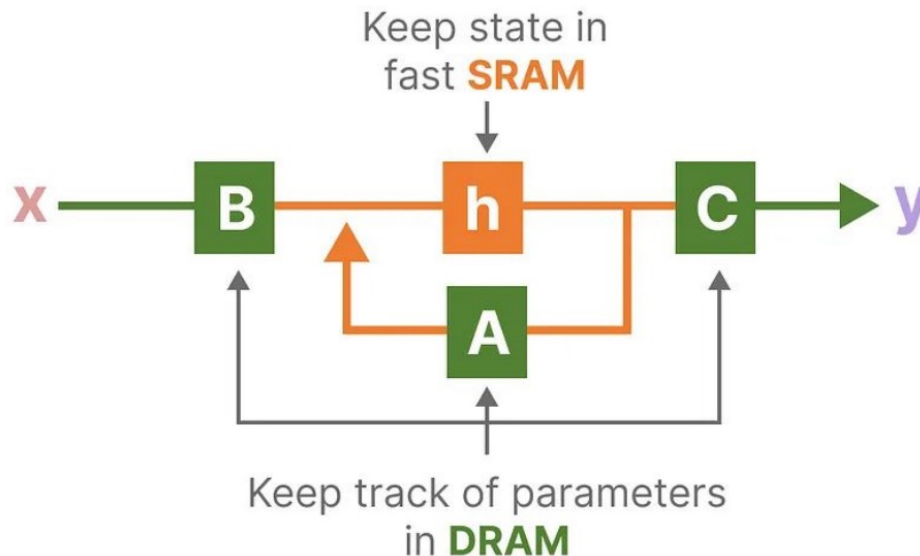
7: **return** y

From S4 to S6: Which Parameters Become Input-dependent?

- S4: A, B, C, Δ are fixed parameters shared across time.
- S6/Mamba: B, C, and Δ depend on x_t , making the model time-varying.
- Convolution no longer applies directly.
- **Hardware-aware computation keeps the expanded state in efficient memory levels.**

Hardware-aware Algorithm

- Mamba utilizes the properties of modern GPUs
 - Restore parallelizability using parallel scan
 - Less memory IO between SRAM and DRAM using kernel fusion and recomputation



The challenge is not only asymptotic complexity; it is also SRAM/DRAM movement on GPUs.

Selective Scan: Making Recurrence Parallel

- The recurrence can be written as an associative composition of affine updates.
- Associativity allows a Blelloch-style scan over sequence positions.
- This restores parallelism even though the model is time-varying.

Prefix Scan (Parallel Scan)

This reduces depth from $O(L)$ to $O(\log L)$ stages.

$$\left(((x_1 \circ x_2) \circ x_3) \circ x_4 \dots \right) = (x_1 \circ x_2) \circ (x_3 \circ x_4) \circ \dots$$

Scan Recovers Parallelizability

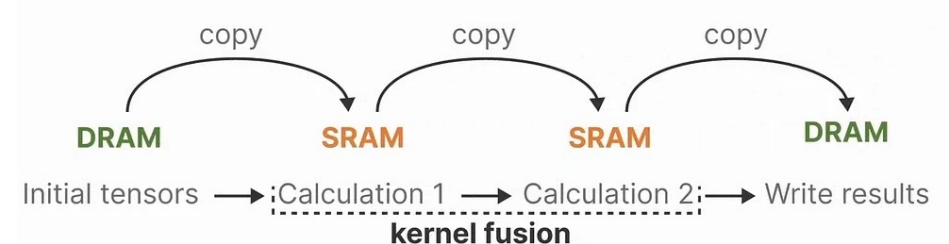
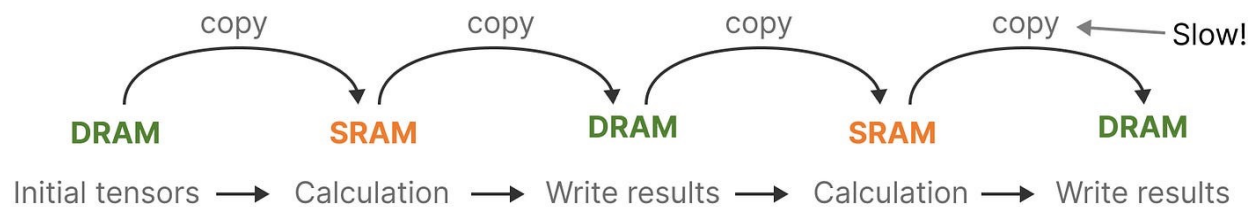
- Without scan recurrent models scale poorly on GPUs.
- With scan SSMs recover much of transformer-style parallel efficiency, while maintaining linear-time complexity.
- That is why Mamba complexity is approximately $O(L)$ instead of transformer attention $O(L^2)$ for sequence length L .

Less Memory IO

- A disadvantage of recent GPUs is their limited transfer (IO) speed between their small but highly efficient SRAM and their large but slightly less efficient DRAM.
 - Frequently copying information between SRAM and DRAM becomes a bottleneck.
- Mamba, like Flash Attention, attempts to limit the number of times we need to go from DRAM to SRAM and vice versa.

Less Memory IO

- Instead of preparing $(\bar{A}, \bar{B})$ in HBM, it loads (A, B, C, Δ) from HBM to SRAM and performs discretization and recurrence in it and writes the final output to HBM
- Steps
 - Read in $O(BLD + DN)$ bytes of memory (Δ, A, B, C) from slow HBM to fast SRAM.
 - Discretize to produce $\bar{A}, \bar{B}$ of size (B, L, D, N) in SRAM
 - Perform a parallel associative scan, yielding intermediate states of size (B, L, D, N) in SRAM.
 - Multiply and sum with C , producing outputs of size (B, L, D) and write it to HBM.



Kernel fusion allows the model to prevent writing intermediate results and continuously performing computations until it is done

Hardware-aware Algorithm: Recomputation

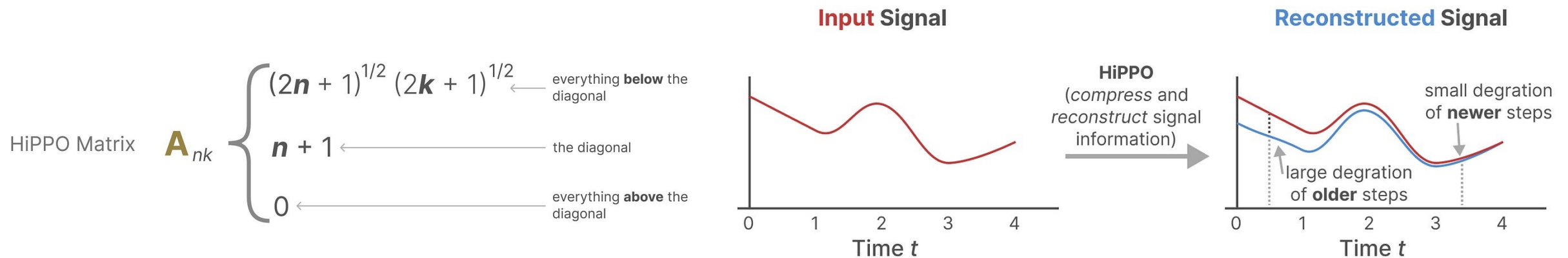
- The intermediate states are not saved but are necessary for the backward pass to compute the gradients. Thus, it recomputes them *during* the backward pass.
 - It does not save intermediate activations that take a lot of memory but are fast to recompute (e.g. output of activation function or short convolution).
- Although this might seem inefficient, it is much less costly than reading all those intermediate states from the relatively slow DRAM.

Selective SSM's Main Contributions

- Recurrent SSM created through discretization
- Selective scan algorithm to selectively compress information
 - which allows the model to filter (ir)relevant information
- Default initialization is S4D which is based on the HiPPO initialization on matrix A to capture long-range dependencies
- Hardware-aware algorithm to speed up computation by efficient storage of (intermediate) results
 - parallel scan
 - kernel fusion
 - recomputation

HiPPo

- **H**igh-order **P**olynomial **P**rojection **O**perators (HiPPo)
- HiPPO was shown to be much better than initializing A as a random matrix.
- It more accurately reconstructs *newer* signals (recent tokens) compared to *older* signals (initial tokens).
 - it produces a hidden state that memorizes its history.
 - by tracking the coefficients of a [Legendre polynomial](#) which allows it to approximate previous history.

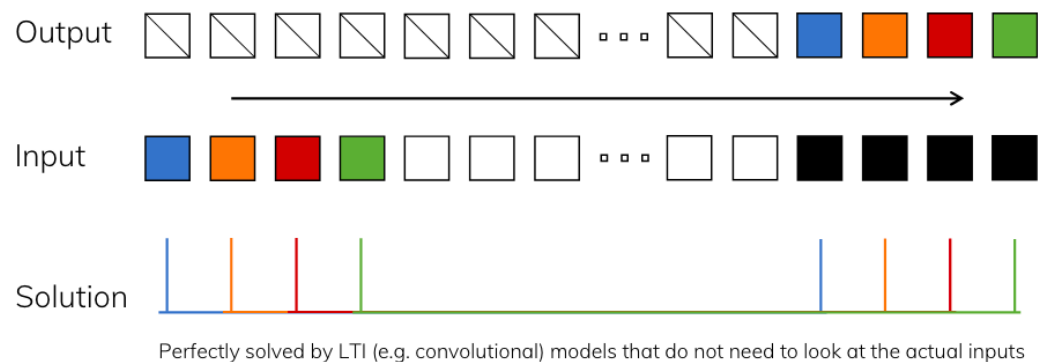


Hardware Awareness: Fusion and Recomputation

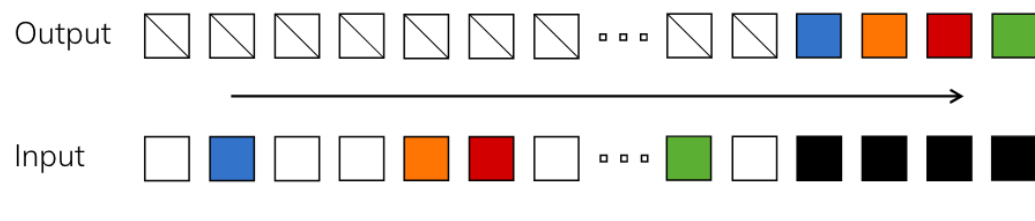
- Keep the recurrent state in fast SRAM as much as possible.
- Avoid writing large intermediate states to HBM/DRAM.
- Use kernel fusion and recomputation to reduce memory IO.
- This systems design is a core part of Mamba, not an implementation footnote.

Synthetic Tasks

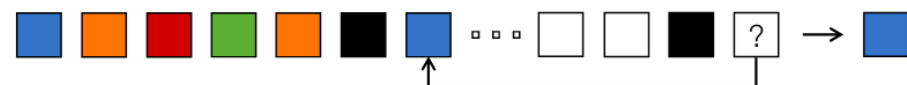
Copying



Selective Copying



Induction Heads



Why Synthetic Tasks Matter?

- Selective copying tests whether the model can remember sparse relevant tokens and ignore distractors.
- Induction heads test pattern matching and extrapolation beyond the training sequence length.
- These tasks expose weaknesses of fixed SSMs and motivate selectivity.

Synthetic-Task Results

Results

- Selection arch(modifying S4 to S6) easily solves selective copying
- Mamba solves induction heads and generalizes perfectly to million-length sequences

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.

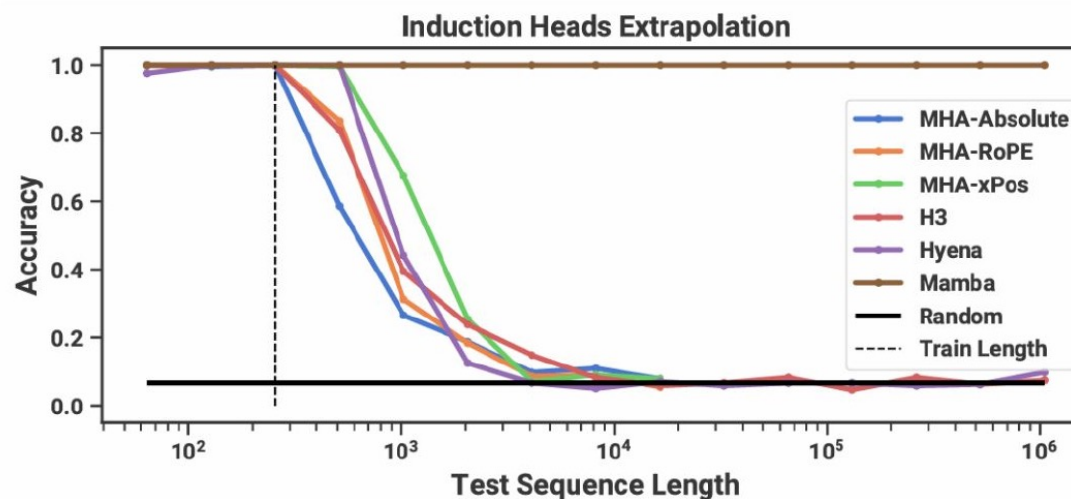


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

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

DNA Scaling Laws

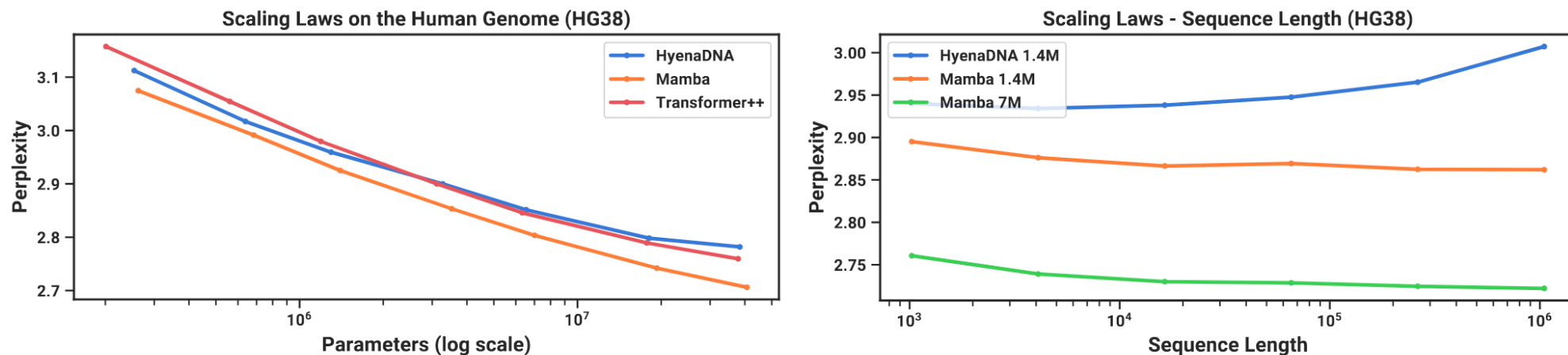


Figure 5: (**DNA Scaling Laws.**) Pretraining on the HG38 (human genome) dataset. (*Left*) Fixing short context length $2^{10} = 1024$ and increasing size from $\approx 200K$ to $\approx 40M$ parameters, Mamba scales better than baselines. (*Right*) Fixing model size and increasing sequence lengths while keeping tokens/batch and total training tokens fixed. Unlike baselines, the selection mechanism of Mamba facilitates better performance with increasing context length.

Speed and Memory Benchmarks

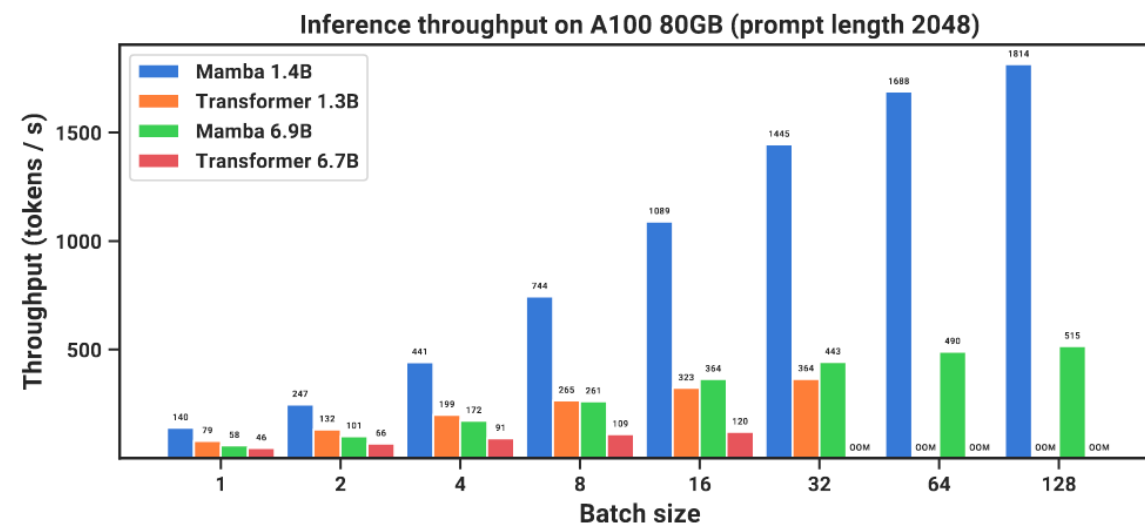
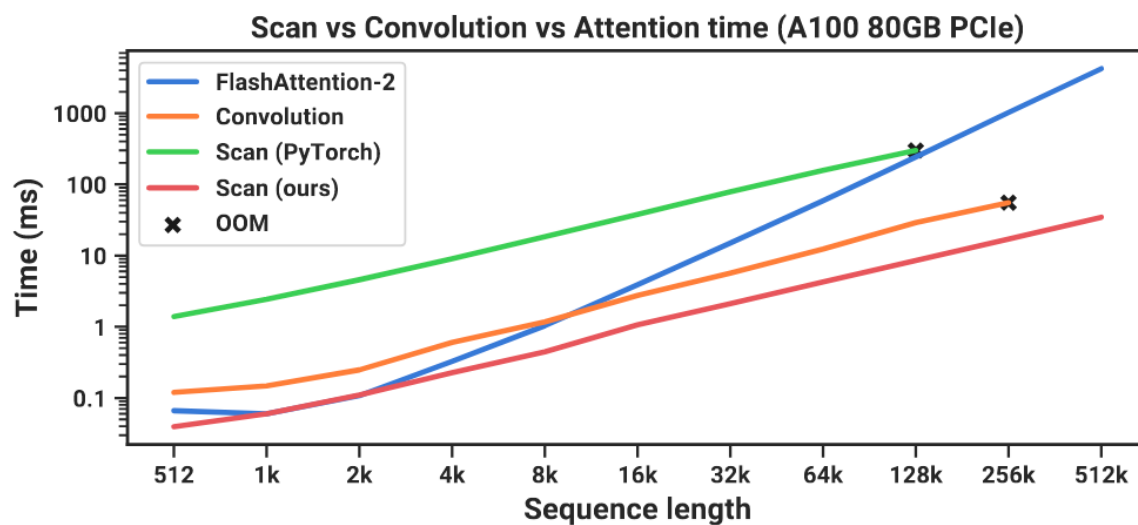


Figure 8: (**Efficiency Benchmarks.**) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

What Mamba Buys You

- Linear-time sequence mixing in the sequence length dimension.
- Constant-size recurrent state at generation time, unlike growing Transformer KV cache.
- Content-dependent routing through selective Δ , B, and C.
- Competitive empirical performance across language, audio, and genomics

Limitations

- Scaling to very large parameter counts requires substantial engineering and specialized kernels.
- Selective dynamics may not always be best for signals where LTI SSMs already provide a strong inductive bias.
- Transformer ecosystems, tooling, and interpretability methods remain much more mature.
- Mamba is not simply a drop-in replacement for attention in every task or hardware regime.

Comparison: attention, RNNs, SSMs, and Mamba

Property	Transformer	RNN	LTI SSM	Mamba/S6
Memory	all tokens / KV cache	hidden state	hidden state	selective hidden state
Training	parallel attention	sequential	convolution/scan	selective scan
Inference	KV grows with length	constant state	constant state	constant state
Content sensitivity	high	medium/high	limited if LTI	high via selection
Core cost	attention over tokens	recurrence	structured recurrence	hardware-aware scan

Summary

- Mamba implements an S6 model: selection plus structured state-space recurrence.
- The key modeling move is input-dependent Δ , B, and C.
- The key systems move is hardware-aware selective scan with fusion and re-computation.
- Mamba is best understood as a recurrent model that restores attention-like content selection.