

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu^{*1} and Tri Dao^{*2}

¹Machine Learning Department, Carnegie Mellon University

²Department 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.


Presentation by Nico Schiavone and Yifeng Chen.

Text Generation

- Amazingly useful task
- Long responses are slow to come out or inaccurate

write a story about watermelons





gemin-2.5
-pro-
exp-03-25

The sun beat down on Elmwood County, turning the asphalt soft and sending shimmering heatwaves rising from the fields. But in Old Man Hemlock's patch, nestled behind his weathered farmhouse, something magical was happening. It was watermelon season.

Hemlock wasn't just a farmer; he was a curator of sweetness, a whisperer to vines. His watermelons weren't merely fruit; they were emerald orbs holding the very essence of summer. He'd planted the seeds back in the cool promise of spring, tiny black teardrops tucked into mounds of rich earth. He'd watched the first tentative sprouts unfurl, then the determined crawl of the vines, their fuzzy tendrils reaching like curious fingers across the warm soil.

Now, the patch was a sprawling kingdom of green. Mottled globes, striped spheres, and oblong giants lay nestled amongst the broad leaves, soaking up the relentless sun. Hemlock, his face shaded by a wide-brimmed straw hat, moved slowly among them. He didn't just look; he listened. He'd tap each melon gently, his ear close to the rind, listening for the deep, resonant *thump* that signaled perfect ripeness – a sound like a tiny drum beating with stored sunshine and sweet water.

There was one melon, near the center of the patch, that he called "The King." It was immense, perfectly round, with dark green stripes cutting sharply against a lighter green background. It had a

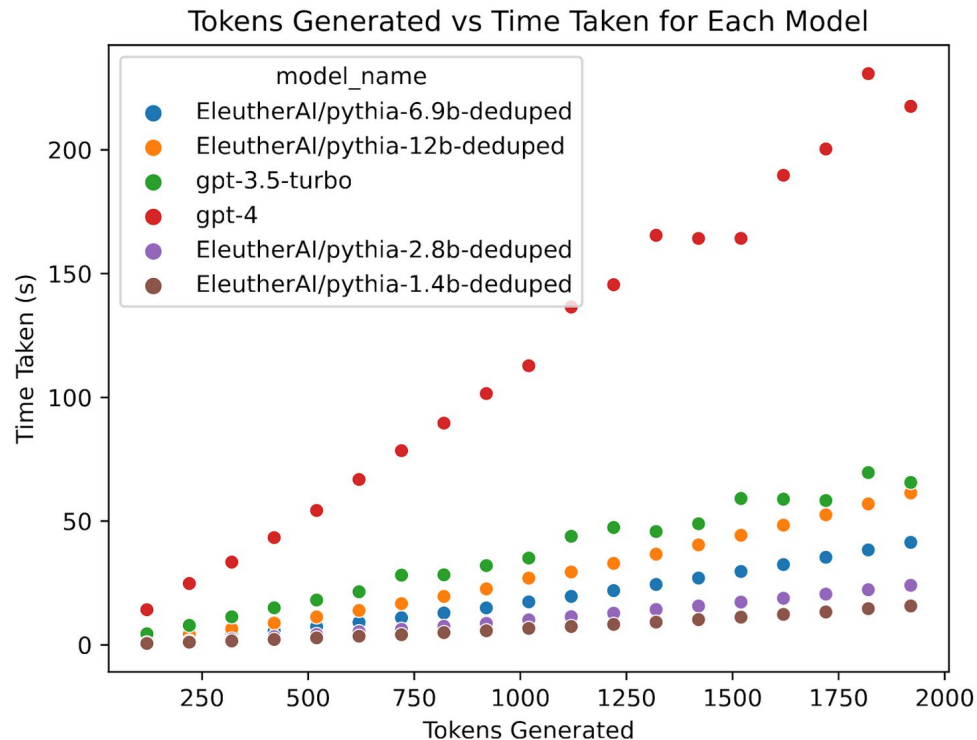
Transformers for Text Generation

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



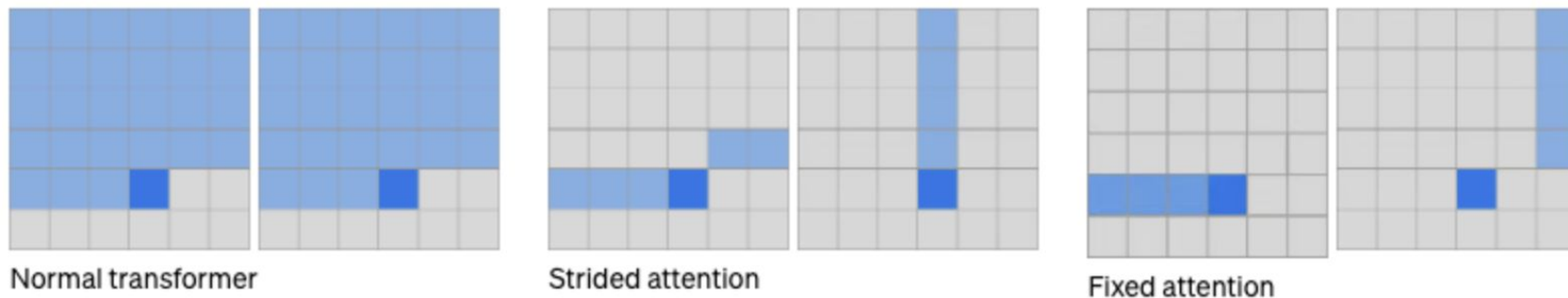
Transformers for Text Generation

Mainly concerned with
long text lengths



Bounded Regret: What Will GPT-2030 Look Like?

Mitigating the Time Complexity Through Speedups



Also IO aware speedup algorithms: FlashAttention, etc.

How can we modify the architecture to reduce inference time complexity?

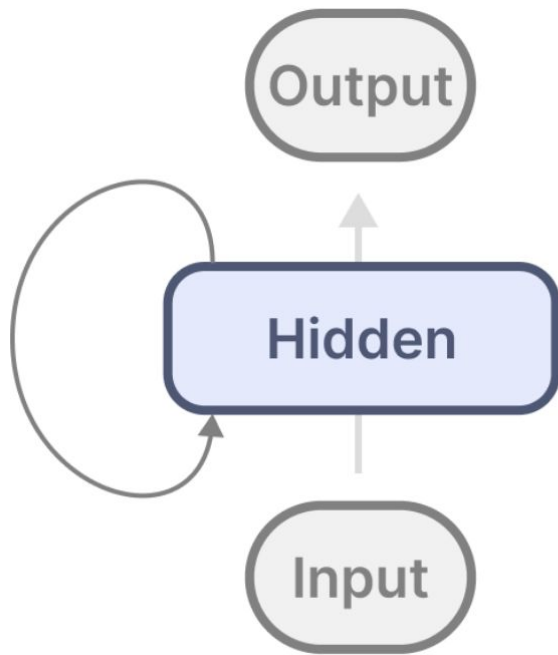
RNNs

Compress information into a hidden state

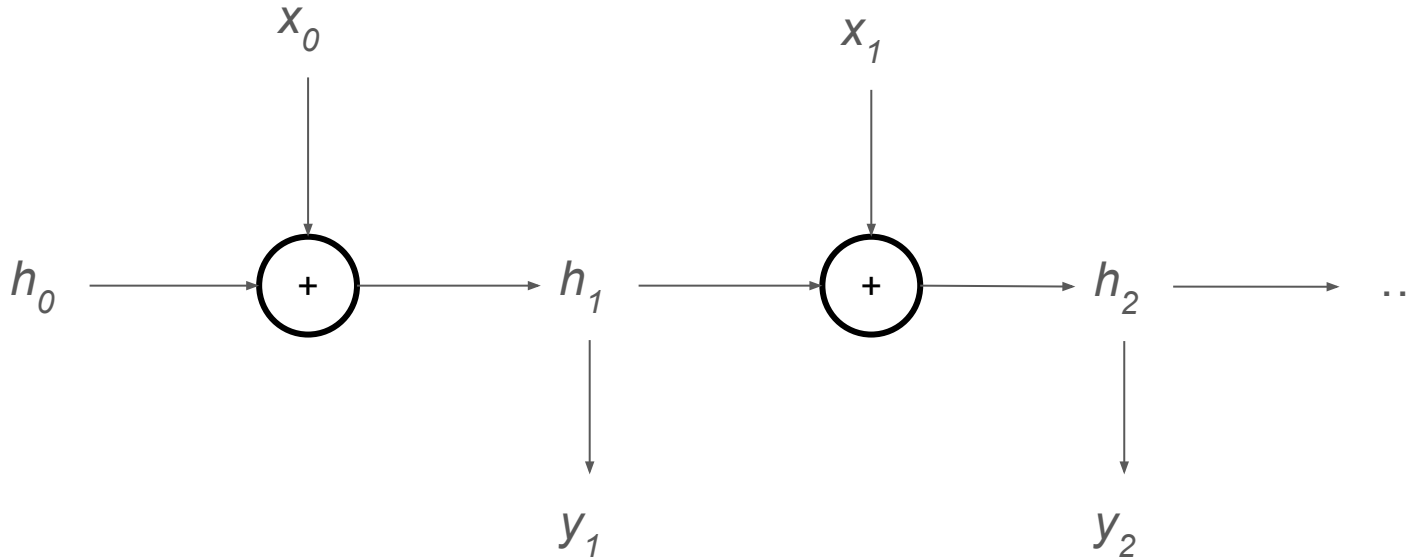
- only care about h and x

Questionable performance on very long data

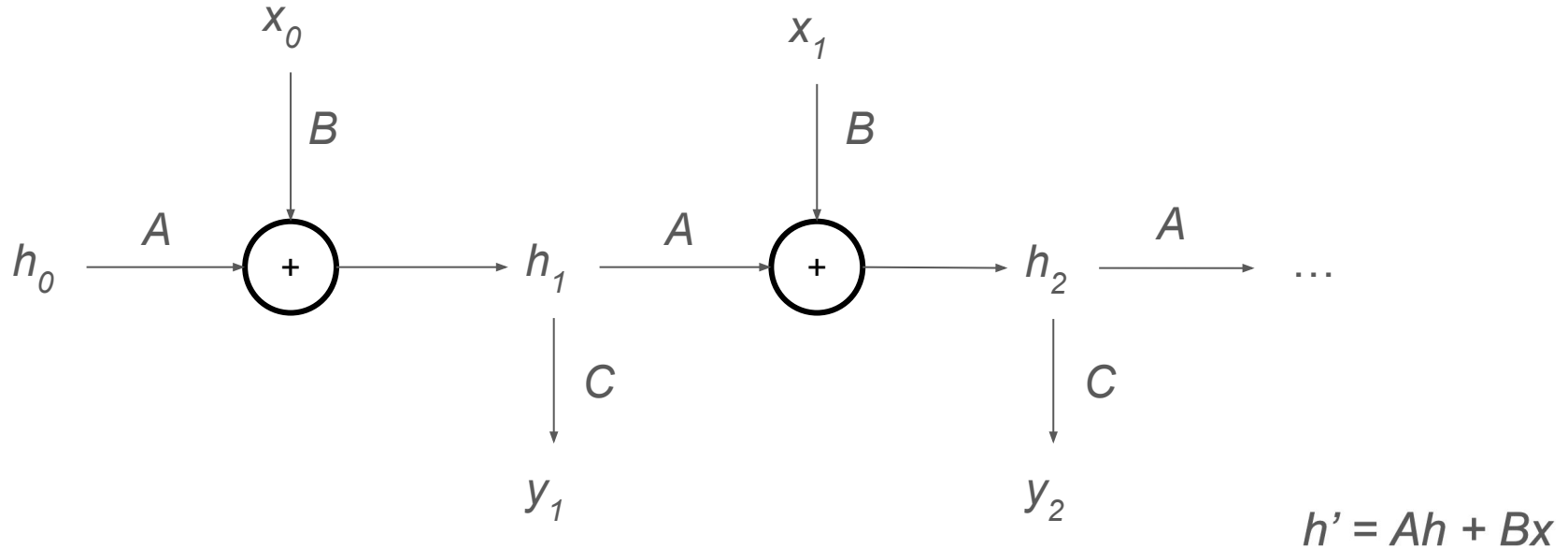
- can't train in parallel



State Space Models: a method made for long sequences



State Space Models: a method made for long sequences



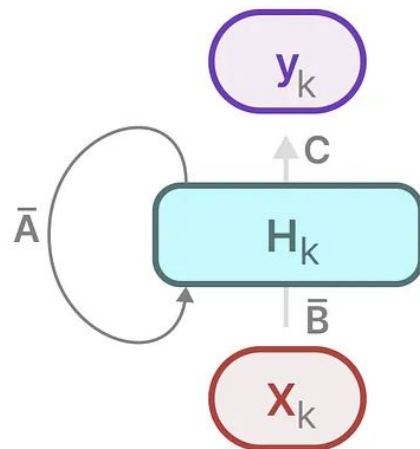
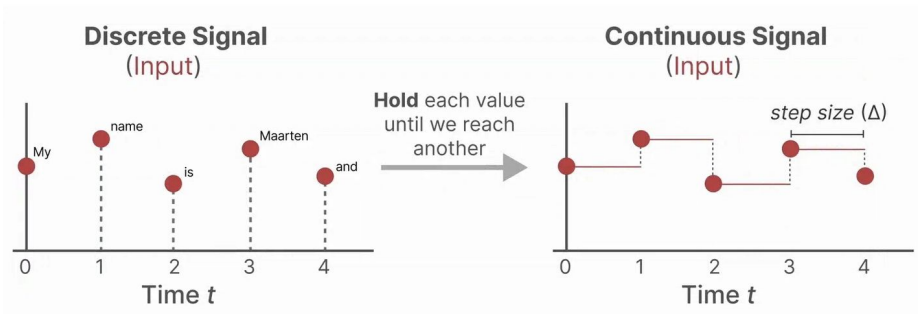
A : how much the state is preserved

B : how much new inputs affect the hidden state

C : converts the hidden state to an output

State Space Models

- Constant time inference
- Must convert discrete data to continuous using Δ
- Parallelizable via convolution due to A, B, C constant



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

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

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \dots, C\bar{A}^{k-1}\bar{B}, \dots)$$

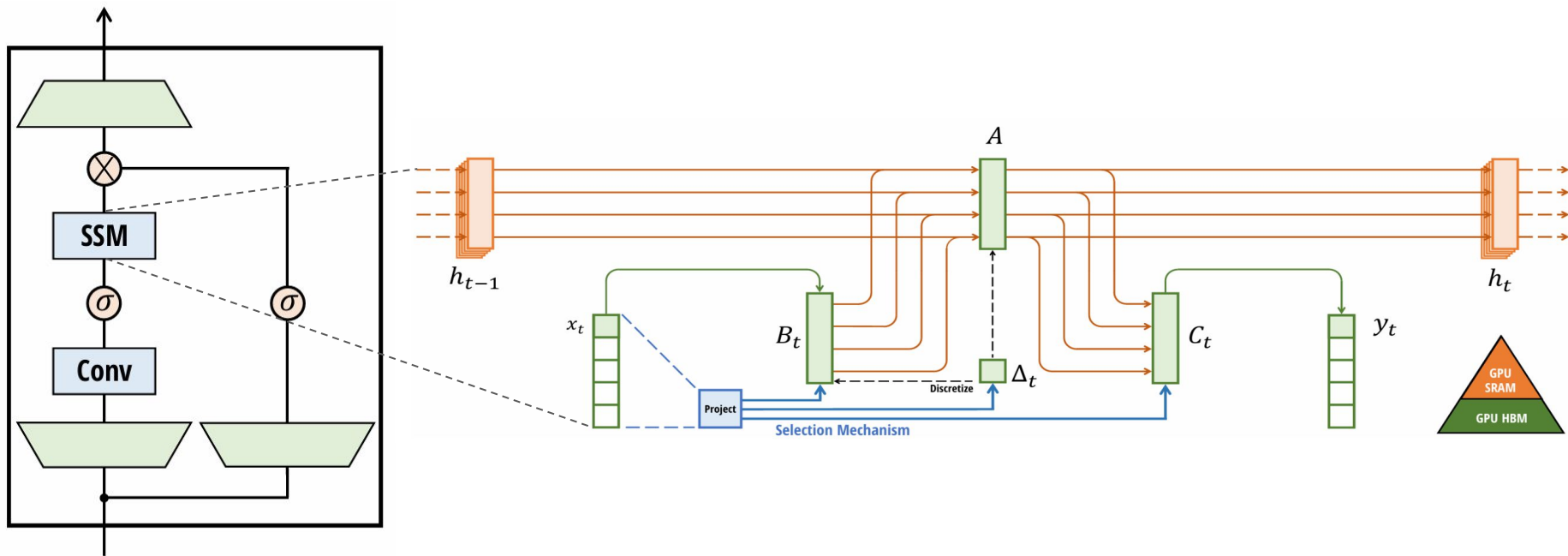
$$y = x * \bar{K}$$

Problems with SSM

- All tokens are compressed into the hidden state, mostly irreversibly
- Requires time invariance, so must treat all tokens equally in the computation of the next state
 - i.e. the parameters are constant

How can we restore the capabilities of transformers while keeping fast inference on long generations?

Mamba



Mamba block and at its core Selective SSM, or S6

Key Idea: Selective state space models

- “The fundamental problem of sequence modeling is *compressing context into a smaller state.*”
- **Selectivity:** the ability to **focus on or filter out** inputs into a sequential state
- A selection mechanism that controls how content affect hidden states(context): **attention in disguise!**

Key Idea: Selective state space models

- By letting parameters matrices that affect interactions along the sequence to be **input-dependent!**
 - Learn linear projectors s_B, s_C, s_Δ to form the matrices from input.
 - Dimension L makes the model time varying.

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

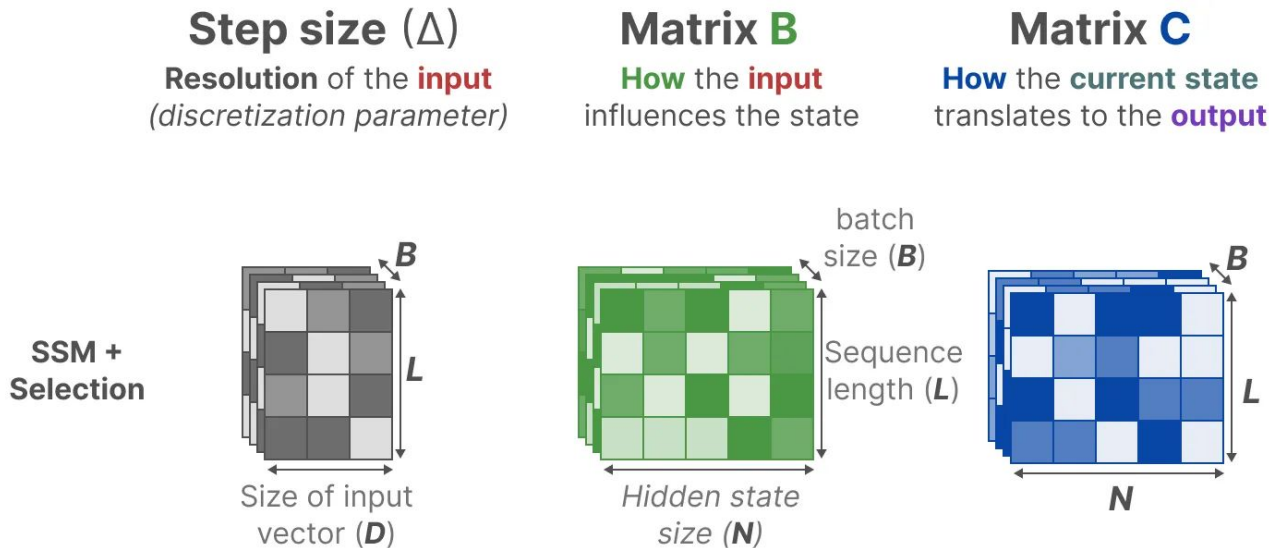
2: $\mathbf{B} : (\mathbf{B}, \mathbf{L}, \mathbf{N}) \leftarrow s_B(x)$ 3: $C : (B, L, N) \leftarrow s_C(x)$ 4: $\Delta : (B, L, D) \leftarrow \tau_{\Delta}(\text{Parameter} + s_{\Delta}(x))$ 5: $\bar{A}, \bar{B} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, A, B)$ 6: $y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$

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

7: **return** y

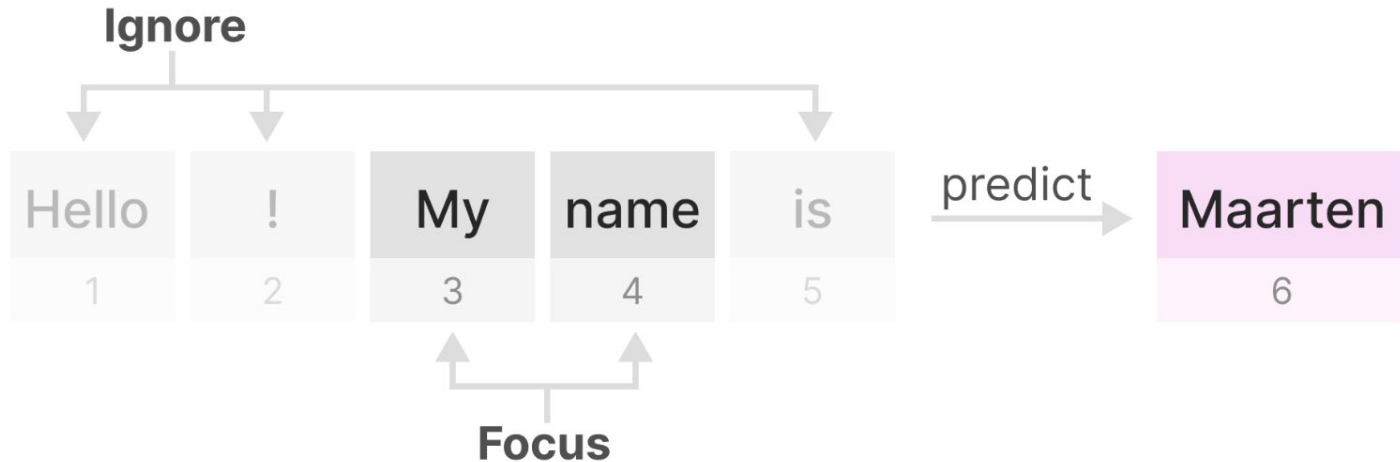
Key Idea: Selective state space models

- Now, for **every input token**, we have different Δ , B , C matrices.
- Together, they **selectively choose** how to modify the hidden state



Key Idea: Selective state space models

- Example of the selective process:

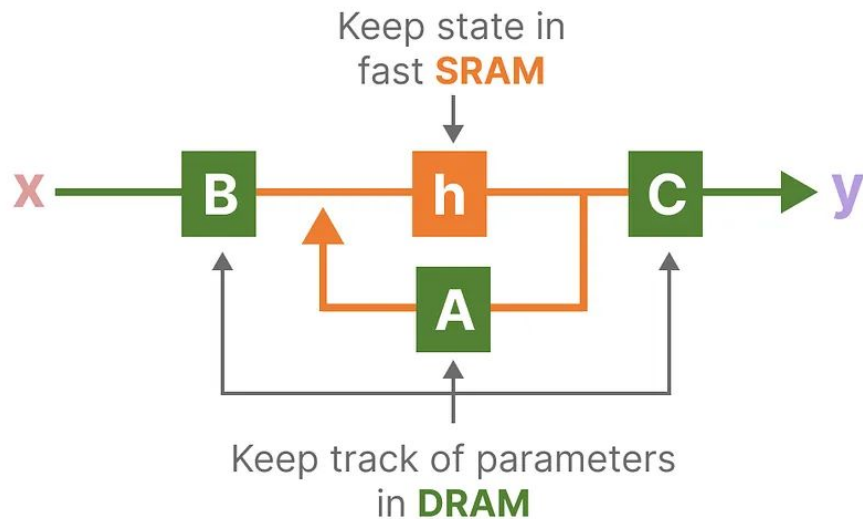


How do we make this still fast?

- When making the model input-dependent, we lose time invariance and the convolution analog.
 - Naive recurrent mode uses $O(BLDN)$ FLOPs. Seems hard to parallel.
 - Prior SSMS use convolution (B,L,D) kernel bypasses the need to materialize (B,L,D,N) hidden states.
- How to implement this selective process **without losing the computation efficiency?**
- Key idea is that we utilize the properties of modern GPUs:
 - Restore parallelizability using **parallel scan**
 - Less memory IOs between SRAM and DRAM using **kernel fusion and recomputation**

Key Idea: Hardware awareness algorithm

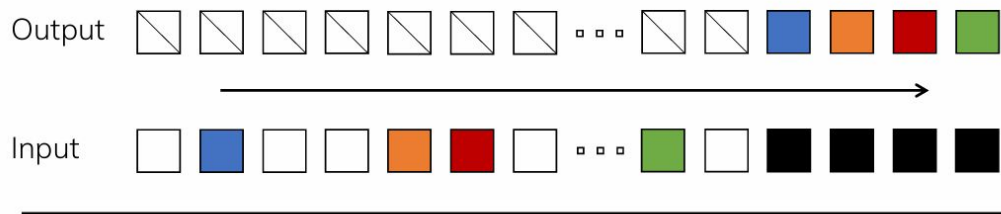
- Uses Classic Blelloch Scan to relieve sequential recurrence
- Reduces copying information between SRAM and DRAM through kernel fusion and recomputation, preventing writing intermediate results



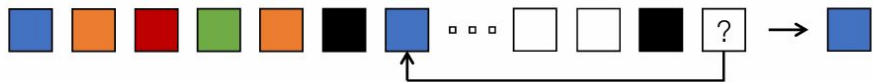
Experiments

- Selective copying
- Induction heads
 - Difficult for state space models

Selective Copying



Induction Heads



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.

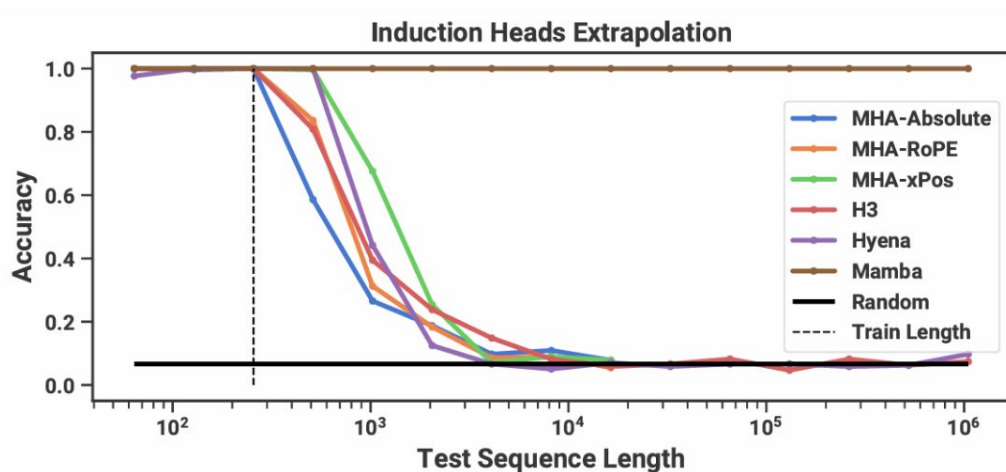


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Limitations

- Untested on higher parameter settings(May involve engineering difficulties)
- Selective architecture may impede performance on data that LTI SSMs excel on (signals).
- No ecosystem yet compared with Transformers.

Summary

- Mamba implements an S6 model (selection + structured state space model)
- Results in linear time inference scaling
- Relies on hardware optimization algorithm

