

# **Structured State-Space Models as Efficient LLM Architectures**

Shicheng Wen



# Why Transformers Struggle

- Self-attention computes all pairwise token interactions
  - $O(T^2)$  time and memory
- When  $T$  is large:
  - GPU memory explodes
  - Training slows dramatically
  - Long-document tasks become impractical



# Do We Really Need Every Token to Compare With Every Other Token?

- In real systems (physical, biological, engineered), we rarely see full pairwise interactions.
- Examples of *state-evolving* systems:
  - Temperature changes
  - Water level in a tank
  - Electrical current
  - Position of a robot arm
- These follow **state-space equations**, not pairwise interactions.

# State-Space Model

- The basic recurrence:  $h_{t+1} = Ah_t + Bx_t$ 
  - $h_t$  = the system's "memory"
  - $x_t$  = input at time t
  - **A** = how memory decays or persists
  - **B** = how input influences the state
- Output:  $y_t = Ch_t$
- Intuition:
  - A controls how long we remember; B controls how strongly inputs affect the system; C reads out the state.

# A Key Insight: Recurrence = Convolution

- Unrolling the recurrence:  $y_t = Ch_t = \sum_{i=0}^t (CA^iB)x_{t-i}$
- This is exactly a **1-D convolution**:

$$y_t = (K * x)_t = \sum_{i=0}^t K_i x_{t-i}$$

- Where  $K_i = CA^iB$
- Convolutions can be computed **in parallel**
- This avoids sequential RNN-style updates
- SSMs achieve  **$O(T)$**  complexity

# How SSMs Capture Long-Range Information

- Let  $A$  have eigenvalues  $\lambda$ .
  - If  $|\lambda| \approx 1 \rightarrow$  memory lasts a long time
  - If  $|\lambda| \ll 1 \rightarrow$  memory fades quickly
- Analogy:
  - $\lambda$  is like the “width of a pipe”:
  - Wider pipe  $\rightarrow$  slow decay  $\rightarrow$  long memory
  - Narrow pipe  $\rightarrow$  fast decay  $\rightarrow$  short memory

# Limitations of Static SSMs

- Despite being efficient, static SSMs have a major issue:
  - **They behave the same way for every input.**
- If we want the model to:
  - treat “because” as a cue for long-range reasoning
  - emphasize certain structures in a sentence
  - adjust behavior based on context
- Static SSMs **cannot adapt.**

# Mamba: Making SSMs Content-Aware

- Mamba allows A, B, and C to depend on the input:

$$A(x_t), \quad B(x_t), \quad C(x_t)$$

- Then the kernel becomes input-dependent:

$$K_i(x) = C(x_t)A(x_{t-1}) \cdots A(x_{t-i})B(x_{t-i})$$

- The model decides what to remember, when to forget, and how to update — based on the content.
  - Similar to attention
  - Still  $O(T)$  complexity

# The Selective Scan Algorithm

- Challenge: Input-dependent SSMs are normally sequential.
- Mamba solves this using a **parallel prefix-style algorithm**:
  - Accumulates dynamic matrices efficiently
  - Fuses operations into one GPU kernel
  - Achieves:
    - $O(T)$  time
    - $O(T)$  memory
    - $O(1)$  autoregressive step

# Mamba2: Theory Meets Practice

- Mamba2 shows something surprising:
- Attention and SSMs are two views of the same structured operator.
- A lower-triangular semiseparable matrix:

$$M_{i,j} = C^T A_j A_{j-1} \cdots A_{j+1} B_i$$

- The computation resembles:  $Y = (L \circ QK^T)V$
- Attention  $\approx$  SSM in another basis
- Deep connection between algorithms

# Complexity Comparison

Model	Training Time	Memory	Autoregressive Step
Transformer (Self-Attention)	$O(T^2)$	$O(T^2)$	$O(T)$
LTI SSM (Static)	$O(T)$ or $O(T \log T)$	$O(T)$	$O(1)$
Mamba (Selective)	$O(T)$	$O(T)$	$O(1)$
Mamba2 (SSD)	$O(T)$ (faster constant)	$O(T)$	$O(1)$

# Summary

- Transformers: expressive but quadratic
- SSMs: efficient, structured, interpretable
- Recurrence  $\leftrightarrow$  convolution duality enables parallelization
- Mamba introduces content selectivity
- Mamba2 provides a theoretical bridge to attention