Mamba

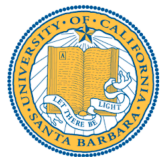
UCSB LLM Seminar

Claire Murphy

UC Santa Barbara

November 20, 2026





Review: The Transformer Architecture

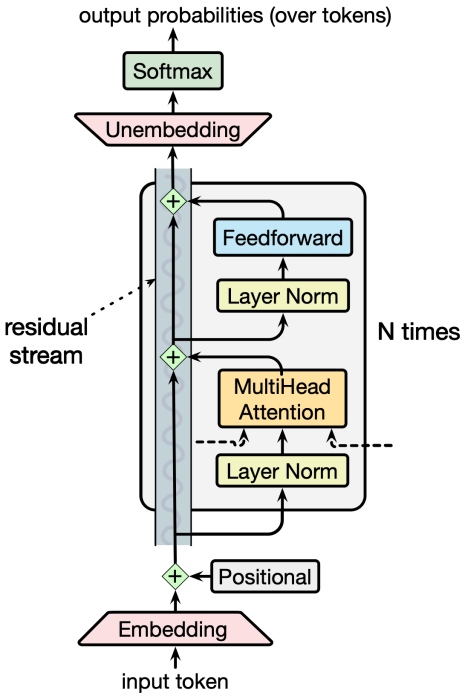


Figure: From [JM].

Autoregressive LLM: Given a prompt, how do we predict the next token in the sequence?

1. Convert the prompt into an $L \times d$ matrix, where each row corresponds to a token.
2. Pass the prompt through several layers of **transformer blocks**; we obtain an $L \times d$ matrix.
   1. Layer norm: Normalizing vectors
   2. Multihead attention: The heart of the Transformer computation
   3. Feedforward layer: Neural network
3. Use the **language modeling head** to produce a probability distribution over all possible tokens. Use this distribution to predict the next token.

Motivation

Self-attention compares every token embedding to every other token embedding. L tokens => $O(L^2)$ complexity.

Previous work: Replace self-attention mechanism with Structured State Space Sequence (S4) mechanism

1. Maps inputs $\{x_1, \ldots, x_L\}$ to outputs $\{y_1, \ldots, y_L\}$ using implicit latent states $\{h_1, \ldots, h_L\}$.
2. Uses fixed time-invariant parameters (Δ, A, B, C).

S4 models "have been successful in domains involving continuous signal data such as audio and vision. However, they have been less effective at modeling discrete and information-dense data such as text."

Solution: (Selection, Scanned) Structured State Space Sequence (S6) mechanism:

1. Compute Δ_i, B_i, C_i from the current input vector, while keeping the learned matrix A fixed across positions.
2. Depending on the current token, Mamba can adjust these parameters to focus on crucial attributes.
3. Hardware-aware parallel scan algorithm overcomes added complexity due to time-dependency.

S4 Mechanism

Mechanism: Map inputs $\{x_1, \ldots, x_L\}$ to outputs $\{y_1, \ldots, y_L\}$ using implicit latent states $\{h_1, \ldots, h_L\}$. To simplify, we assume each $x_i, y_i \in \mathbb{R}$. Each $h_i \in \mathbb{R}^N$.

1. Parameters:

Δ	A	B	C
1×1	$N \times N$	$N \times 1$	$1 \times N$

2. Discretization: Fix a **discretization rule** (f_A, f_B). Define

$$\overline{A} = f_A(\Delta, A) \in \mathbb{R}^{N \times N}, \quad \overline{B} = f_B(\Delta, A, B) \in \mathbb{R}^{N \times 1}.$$

3. Computation: Two methods
   1. Linear recurrence: Used when input vectors are seen one at a time.

$$h_i = \overline{A} h_{i-1} + \overline{B} x_i, \quad y_i = C h_i \overset{h_0=0}{=} \sum_{j=1}^{i} C(\overline{A})^{i-j} \overline{B} x_j.$$

   2. Global convolution: Used for parallelization. Define

$$\overline{K} = (C\overline{B}, C\overline{AB}, C(\overline{A})^2\overline{B}, \ldots, C(\overline{A})^{L-1}\overline{B}).$$

$y_i = (x * \overline{K})_i$.

Some Notes about S4

Structure: Efficient computation of the S4 mechanism requires imposing additional structure on A. The most popular structure is diagonal.

S4 models use the S4 mechanism for sequence mixing in place of attention. The S4 mechanism uses **input-independent** sequence-mixing coefficients. Attention instead computes input-dependent mixing coefficients, while also sharing its learned projection matrices across positions.

Examples: Linear attention, H3 (most popular), Hyena, RetNet, etc.

Where S4 Models Go Wrong

The authors use two synthetic tasks to motivate input-dependent selection:

1. Selective Copying
   1. Input: Colored tokens, interspersed with white tokens. At the end of the input, there are black tokens.
   2. Output: The colored input tokens, in the correct order.
2. Induction Heads
   1. Input: A sequence of colored tokens.
   2. Output: The predicted next token in the sequence.

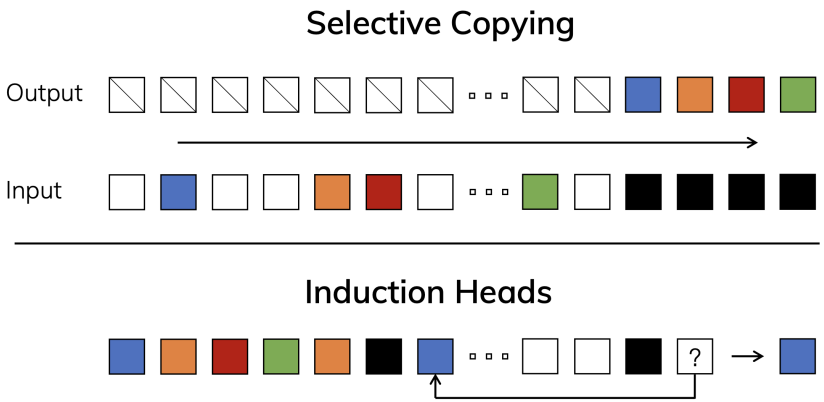


Figure: When the input length is longer than the training input length, the performance of S4 models begins to deteriorate.

Why does S4 Go Wrong?

Two major limitations of S4:

1. Input-independent weights
   - S4 mechanism: $y_i = \sum_{j=1}^{i} C(\overline{A})^{i-j}\overline{B}x_j$.
   - Regardless of the value of x_j, the weight of this jth token remains the same.
   - How does this relate to the two tasks above?
     1. Selective copying: Only colored tokens should affect token prediction.
     2. Induction heads: If we want to predict the token appearing after a black token, previous "after-black" tokens should be weighted more heavily.
   - This is in sharp contrast to attention: $y_i \approx \sum_{j=1}^{i} \langle Qx_i, Kx_j \rangle Vx_j$.
2. Information compression
   - S4 mechanism: $h_i = \overline{A}h_{i-1} + \overline{B}x_i$, $y_i = Ch_i$.
   - $h_i \in \mathbb{R}^N$ must contain all relevant information about $\{x_1, \ldots, x_{i-1}\}$. If $i \gg N$, we lose valuable information.
   - In the context of the two tasks above, if patterns are “too long” to fit within some $h \in \mathbb{R}^N$, we are unable to correctly predict the next token.
   - Again, self-attention avoids information compression.

Note: Information compression can also be thought of as a **strength** of the S4 mechanism, because it requires less memory and is less computationally complex than self-attention.

S6 Mechanism

As in S4, we assume $x_i, y_i \in \mathbb{R}$. Each $h_i \in \mathbb{R}^N$.

1. Selection: Keep diagonal A fixed across positions. Compute coefficients from the input x_i:

$$\underset{1\times 1}{\Delta_i = \text{softplus}\big(\delta + s_\Delta(x_i)\big)} \qquad \underset{N\times N}{A} \qquad \underset{N\times 1}{B_i = s_B(x_i)} \qquad \underset{1\times N}{C_i = s_C(x_i)}$$

   Here, $\text{softplus}(x) = \ln(1 + e^x)$. Learn A, s_B, s_C, s_Δ and δ.

2. Discretization: At each position,

$$\overline{A}_i = f_A(\Delta_i, A), \qquad \overline{B}_i = f_B(\Delta_i, A, B_i).$$

3. Computation: Use the time-varying recurrence

$$h_i = \overline{A}_i h_{i-1} + \overline{B}_i x_i, \qquad y_i = C_i h_i.$$

There is generally no single time-invariant convolution kernel.

Efficient Implementation of S6

The S6 mechanism loses the fixed-convolution computation.

To overcome this limitation, the authors pursue the following three strategies:

1. Parallel scan: For a known input sequence, define the affine updates

$$F_i(h) = \overline{A}_i h + \overline{B}_i x_i \Rightarrow h_i = F_i(h_{i-1}) = (F_i \circ \cdots \circ F_1)(h_0).$$

   For $1 \leq i \leq L$, compute $(F_i \circ \cdots \circ F_1)$, then compute $h_1, \ldots, h_L$ in parallel.

2. Kernel fusion: Combine discretization, the scan, and the readout $y_i = C_i h_i$ into one GPU routine. In other words, complete the entire S6 mechanism on SRAM (small, fast on-chip memory), then write output to HBM (larger GPU memory).

3. Recomputation: If hidden state calculations are not stored in HBM, how do we backpropagate in training? Recalculating these hidden states for the gradient calculation is cheaper than saving them in HBM.

Mamba Architecture

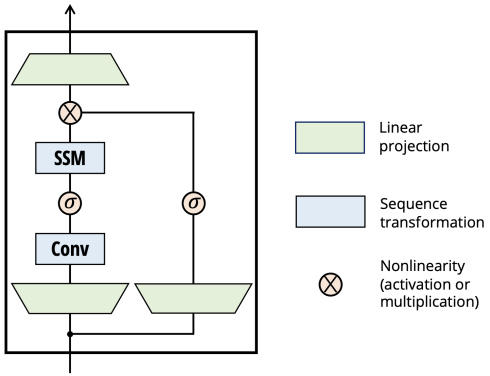


Figure: From [GD].

A block within the Mamba architecture.

1. Green trapezoids: Linear projection, i.e. $u_i = Wx_i$, where W is a learned linear map.
2. "Conv": Combine information from nearby tokens. For example,

$$z_i = [\text{Conv}(u_1, \ldots, u_L)]_i = w_0 u_i + w_1 u_{i-1} + w_2 u_{i-2}.$$

   Here, w_0, w_1, and w_2 are learned weights.
3. σ: Nonlinear activation. In practice,

$$\sigma(a) = \text{SiLU}(a) = \frac{a}{1 + e^{-a}},$$

   applied separately to every coordinate.
4. $\otimes$: Coordinate-wise multiplication.

Synthetic Tasks: Experiments

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.

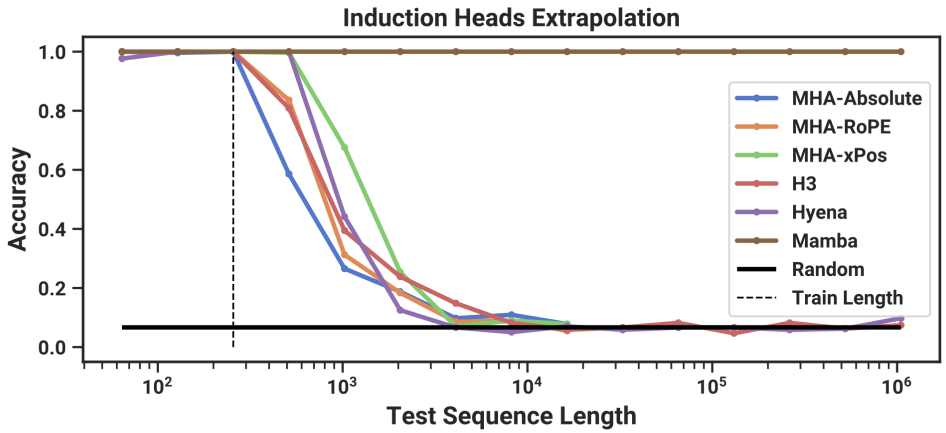


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Figure: Figures from [GD].

Language Modeling: Experiments

Task: Token prediction.

Perplexity $\sim$ (probability the model assigns to the CORRECT next token)$^{-1}$.
Lower perplexity $\Rightarrow$ better model.

FLOPs: Floating-point operations. Number of operations required to train the model.

Transformer++: "The strongest Transformer recipe we know of."

"Mamba is the first attention-free model to match the performance of a very strong Transformer recipe (Transformer++) that has now become standard, particularly as the sequence length grows."

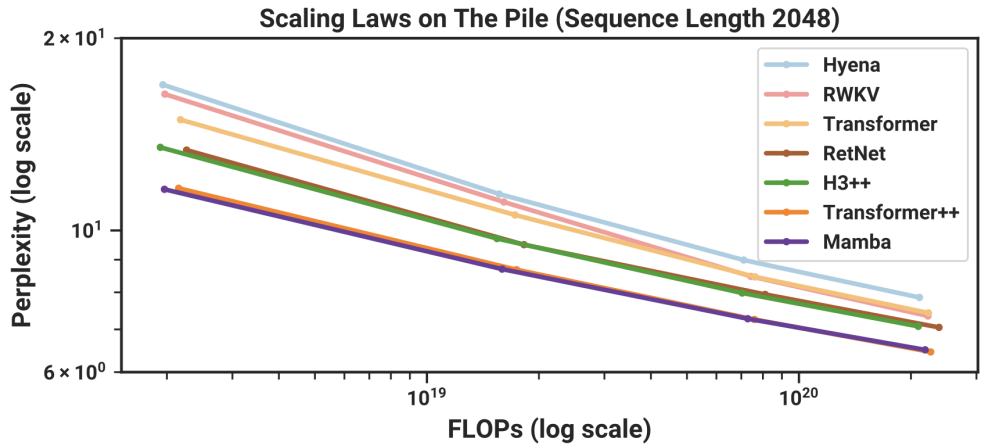


Figure: From [GD].

Conclusion

Self-attention is extremely accurate, but can be computationally costly.

On the other hand, the **S4 mechanism** is less computationally complex and performs well in domains that process continuous (audio or visual) signal data.

S4 models do **not** consider input when weighting previous tokens, which makes them poorly suited to tasks such as selective copying or induction heads.

The **S6 mechanism** allows different weights for different inputs. Scanning and related techniques reduce the computational complexity of this mechanism.

The **Mamba architecture** utilizes this S6 mechanism to successfully complete a range of tasks, including selective copying, induction heads, and language modeling.

Fun fact: It's called Mamba because "SSSSSS" (S6) sounds like the hiss of a snake.

References

GD Gu, Albert and Tri Dao. “Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” 2024.

JM Jurafsky, Daniel and James H. Martin. “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models,” 3rd edition draft, 2026.