A Brief Overview of Recent Advances in Sequence Modeling Through Structured State Space Models

Seminar I

Jyotirmaya Shivottam

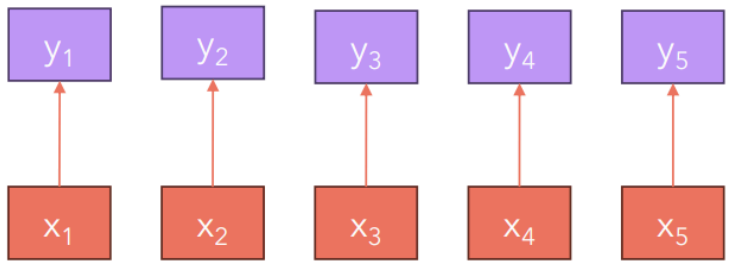
23226001

School of Computer Sciences

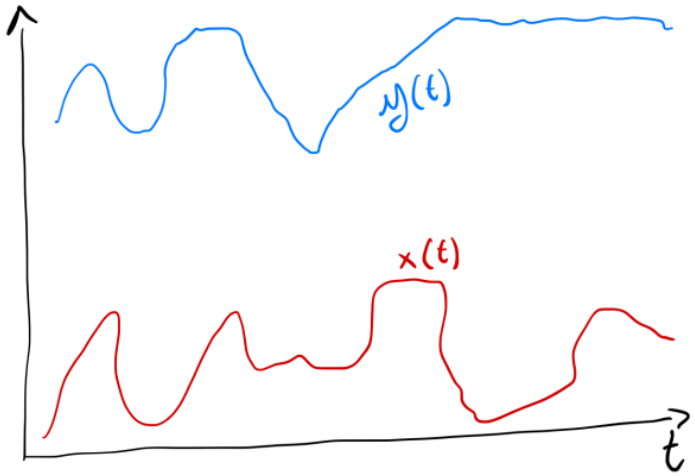
March 04, 2024

Sequence Modeling

- The goal here is to find (or learn) a mapping (model) from a sequence of input data to a sequence of output data.



Discrete signal



Continuous signal

Source: https://github.com/hkproj/mamba-notes

- Some examples include: Time Series Forecasting, Speech Recognition, Music Generation, Language Translation, and Protein Folding.

Sequence Modeling - Common Approaches

Model	Training Complexity	Inference Complexity	Benefits	Issues
Recurrent Neural Networks	$\mathcal{O}(N)$	$\mathcal{O}(1)$	Theoretically infinite context window	Vanishing gradients, Long-term dependencies, Harder to parallelize
Convolutional Neural Networks	$\mathcal{O}(K)$	$\mathcal{O}(K)$	Highly parallelizable	Limited receptive field ($\sim$ context window)
Transformer Networks	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	Highly parallelizable, Full context	Does not scale well for long context windows $\sim$ sequence length

- Here, N is the sequence length, while K is the kernel size.
- Ideal scenario: $\mathcal{O}(N)$ training and $\mathcal{O}(1)$ inference + full context.

Higher Order Polynomial Projection Operators

- HiPPO [1] is a novel recurrent sequence modeling approach that uses polynomial basis projection with a decay measure to model long sequences.
- The idea is to **maintain precise memory** (reconstruction) of the **recent past**, while **forgetting** the **distant past** (as controlled by the decay measure).

Source: MedAI #41: Efficiently Modeling Long Sequences with Structured State Spaces | Albert Gu

HiPPO: Deriving the Update

1 Pick the measure

2 Choose a basis

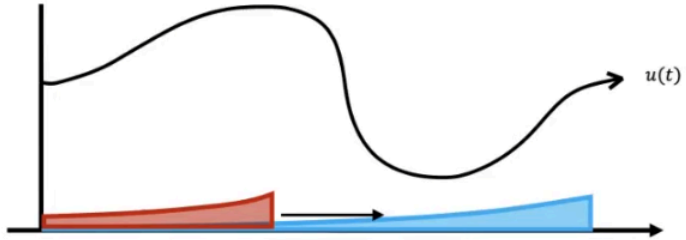
$$x_i(t) = \langle u_{\leq t}, v_i^{(t)} \rangle = \int_0^t u(x) v_i^{(t)}(x) \mu^{(t)}(x)\, dx$$

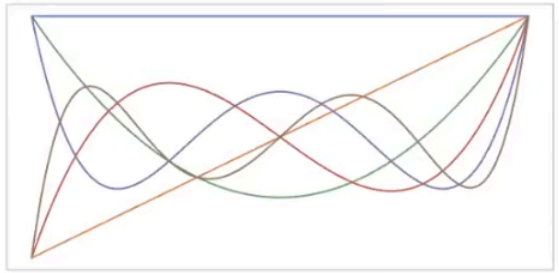
3 State dynamics $x(t)$

Differentiate through the approximation integral

$$x_i'(t) = \int_0^t u(x) \underbrace{\left(\frac{\partial}{\partial t} v_i^{(t)}(x) \right)}_{\sum_j v_j^{(t)}(x)} \mu^{(t)}(x)\, dx + \int_0^t u(x) v_i^{(t)}(x) \left(\frac{\partial}{\partial t} \mu^{(t)}(x) \right) dx$$

$$= \sum_j A_{ij} x_j(t) + B_j u(t)$$





Orthogonal polynomials

Source: MedAI #41: Efficiently Modeling Long Sequences with Structured State Spaces | Albert Gu

- The readout is given by another transform: $y(t) = f(x(t), u(t)) \equiv Cx(t) + Du(t)$.

State Space Models

- A state space model maps an input signal, $x(t)$, to an output signal, $y(t)$, through a set of **hidden state variables, $h(t)$**, like so:

$$\dot{h}(t) = Ah(t) + Bx(t) \leftarrow \text{First order DE}$$
$$y(t) = Ch(t) + Dx(t)$$

- Here, A, B, C, and D are the state **transition**, input, output, and feed-through matrices, respectively. Also, *cf. the HiPPO update equations*.
- SSMs are widely used in control systems, signal processing, and time series analysis.
- They are particularly useful for **linear, time-invariant (LTI) systems**.
- We usually work with discrete data (via sampling), so we need to **discretize** the model.

State Space Models - Discretization

- Discretization is the process of converting a continuous-time model to a discrete-time model.
- The most common method used in modern SSMs (cf. HiPPO [1] & S4 [2]) is the **zero-order hold (ZOH)** method, which arises naturally as the exact solution to the SSM ODEs:

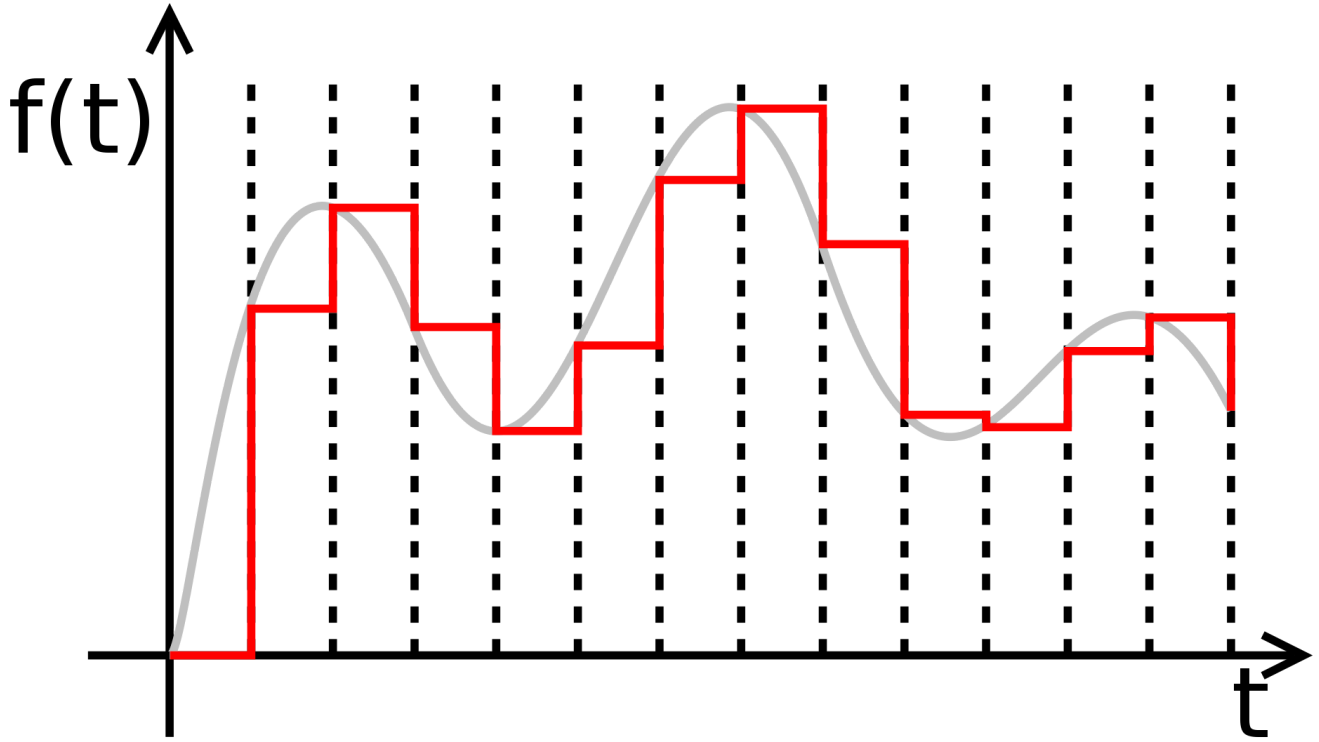
$$h(t) = e^{At}h(0) + \int_0^t e^{A(t-\tau)}Bx(\tau)d\tau$$

$$h[k+1] = e^{A\Delta t}h[k] + \int_k^{k+1} e^{A(k+1-\tau)}Bx(\tau)d\tau$$

$$\equiv h[k+1] = \bar{A}h[k] + \bar{B}x[k], \text{ where, } \bar{A} = e^{\Delta A}, \bar{B} = (\Delta A)^{-1}(e^{\Delta A} - I)(\Delta B)$$

- ZOH is equivalent to **sampling** the continuous-time model at regular intervals, and **holding the input constant** between samples.

State Space Models - Discretization



Source: https://en.wikipedia.org/wiki/Zero-order_hold#/media/File:Zeroorderhold.signal.svg

Structured State Space (Sequence) Models (S4)

- First proposed in [2], these SSMs impose a structure on the state transition matrix, A (or $\bar{A}$).
- This structure is usually **block-diagonal** or **block-triangular**, with specific entries. This allows for **efficient computation** of the state transition.
- A (or $\bar{A}$) is responsible for capturing the information from the input sequence and maintaining the temporal dependencies (i.e., the accuracy of reconstruction), *especially over very long sequence lengths* $\longrightarrow$ **a good initialization for A is paramount**.
- The HiPPO theory provides one such initialization for A.

Structured State Space (Sequence) Models (S4)

2.2 Addressing Long-Range Dependencies with HiPPO

Prior work found that the basic SSM (1) actually performs very poorly in practice. Intuitively, one explanation is that linear first-order ODEs solve to an exponential function, and thus may suffer from gradients scaling exponentially in the sequence length (i.e., the vanishing/exploding gradients problem (Pascanu et al., 2013)). To address this problem, the LSSL leveraged the HiPPO theory of continuous-time memorization (Gu et al., 2020a). HiPPO specifies a class of certain matrices $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ that when incorporated into (1), allows the state $x(t)$ to memorize the history of the input $u(t)$. The most important matrix in this class is defined by equation (2), which we will call the HiPPO matrix. For example, the LSSL found that simply modifying an SSM from a random matrix $\boldsymbol{A}$ to equation (2) improved its performance on the sequential MNIST benchmark from 60% to 98%.

$$\textbf{(HiPPO Matrix)} \qquad \boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}. \tag{2}$$

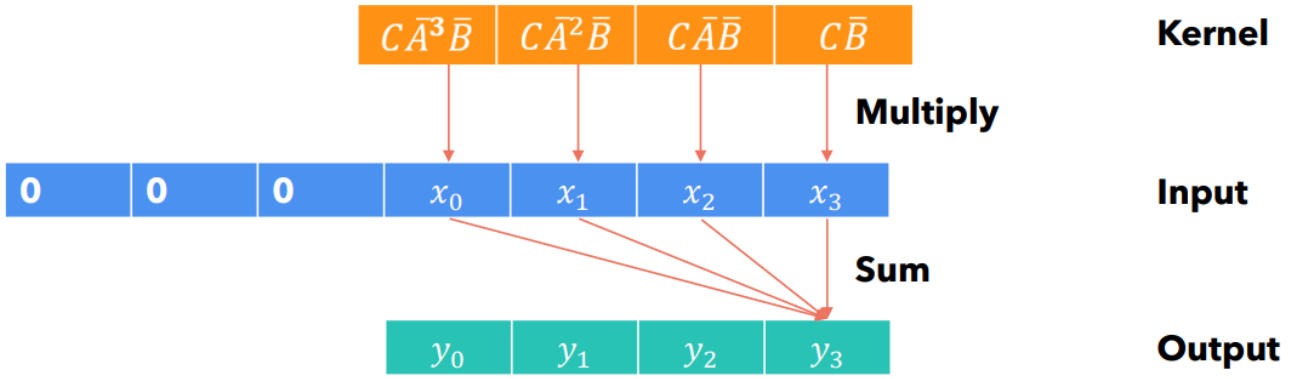
Source: S4, Gu et al [2]

Structured State Space (Sequence) Models (S4)

- Alongside the initialization, the authors also propose a fast computation method for the SSM updates, that leverages the LTI nature of the model.
- Since A, B, C, and D are all **time-invariant (i.e., independent of current input)**, the recurrence can be unrolled to a set sequence length (context window), giving a **fixed convolutional kernel**:

$$
\begin{aligned}
h[k+1] &= \bar{A}h[k] + \bar{B}x[k] \text{ and } y[k] = Ch[k] \\
h[0] &= \bar{B}x[0] \text{ and } y[0] = Ch[0] \\
\implies h[1] &= \bar{A}(\bar{B}x[0]) + \bar{B}x[1] \text{ and } y[1] = C(\bar{A}(\bar{B}x[0]) + \bar{B}x[1]) \\
&\vdots \\
\implies h[k] &= \bar{A}^k h[0] + \bar{A}^{k-1}\bar{B}x[1] + \bar{A}^{k-2}\bar{B}x[2] + \ldots + \bar{B}x[k] \text{ and} \\
y[k] &= C\bar{A}^{k-1}\bar{B}x[0] + \cdots + C\bar{B}x[k] \\
\implies y[k] &= x * \bar{K}, \text{ where} K = (C\bar{A}^{k-1}\bar{B}, \ldots, C\bar{B}) \text{ is the convolution kernel.}
\end{aligned}
$$

Structured State Space (Sequence) Models (S4)



$$y_3 = C\bar{A}^3\bar{B}x_0 + C\bar{A}^2\bar{B}x_1 + C\bar{A}\bar{B}x_2 + C\bar{B}x_3$$

Source: https://github.com/hkproj/mamba-notes

- Two key takeaways:
  i. $\bar{K}$ makes the S4 computation fast (parallelizable over input), while the recurrent mode (SSM equations) allow for fast inference, i.e., training is $\mathcal{O}(N)$, and inference is $\mathcal{O}(1)$.
  ii. HiPPO initialization of A allows calculation of A^k to be stable (to some extent) for much larger k ($>$ 2-3 orders) as compared to SOTA Transformers.

Selective Structured State Space (Sequence) Model with Associative Scan ≡ SSSSSS (S6) ≡ Mamba

- Mamba [3] *relaxes the time-invariance of S4*, to make the model more expressive on certain tasks. However, $\bar{K}$ **is not fixed anymore**. So, the authors use **associative parallel scan** to make the recurrence computation fast.
- It leverages the discretization cleverly to serve as **a gating (or selection) mechanism** over input channels.
- Mamba is the current state-of-the-art SSM approach to sequence modeling, attaining great scores on long range tasks (i.e., where long context is needed).

Mamba (S6): Motivation

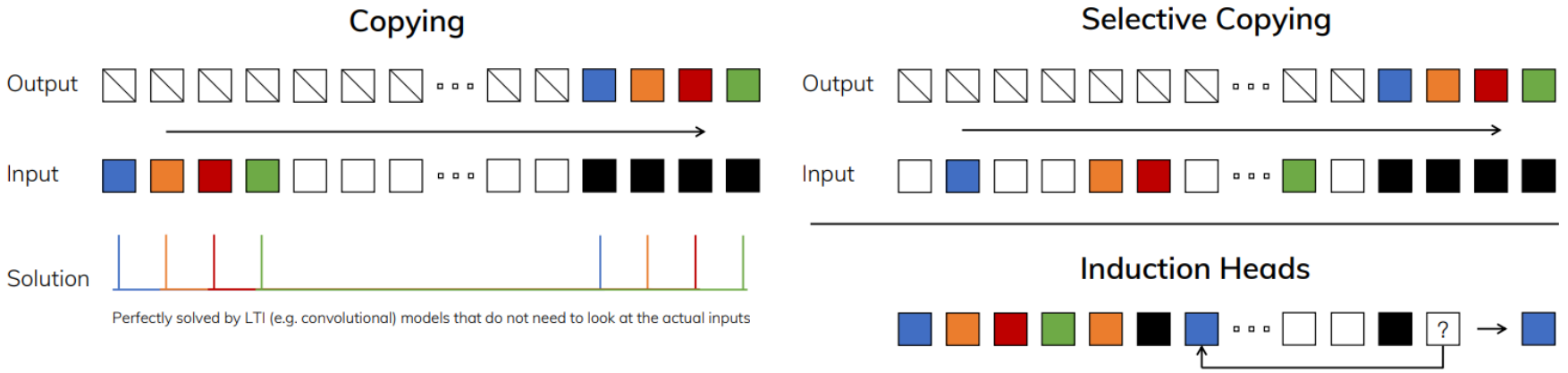


Figure 2: (*Left*) The standard version of the Copying task involves constant spacing between input and output elements and is easily solved by time-invariant models such as linear recurrences and global convolutions. (*Right Top*) The Selective Copying task has random spacing in between inputs and requires 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.

Source: Mamba (S6), Gu et al [3]

- LTI models like S4 face difficulties on these tasks, as their **constant dynamics (A & B are constant)** do not factor in the **current context (i.e., input)**. The inherent **shift-invariance** becomes problematic.

Mamba (S6): A Time-Varying SSM

- Relaxing the LTI criterion: A, B, and C are now time-variant. A is still structured.

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
3: $\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter
4: $\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\text{Parameter})$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow \text{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \text{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
7: **return** y

$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$
$$y_t = C h_t$$

The $\bar{A}$ matrix also depends on the input, through Δ

Algorithm 2 SSM + Selection (S6)

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

$$s_B(x) = \text{Linear}_N(x),\ s_C(x) = \text{Linear}_N(x),\ s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x)),\ \text{and}\ \tau_\Delta = \text{softplus}$$

Source: Mamba (S6), Gu et al [3] & https://github.com/hkproj/mamba-notes

Mamba (S6): Gating or Selection Mechanism

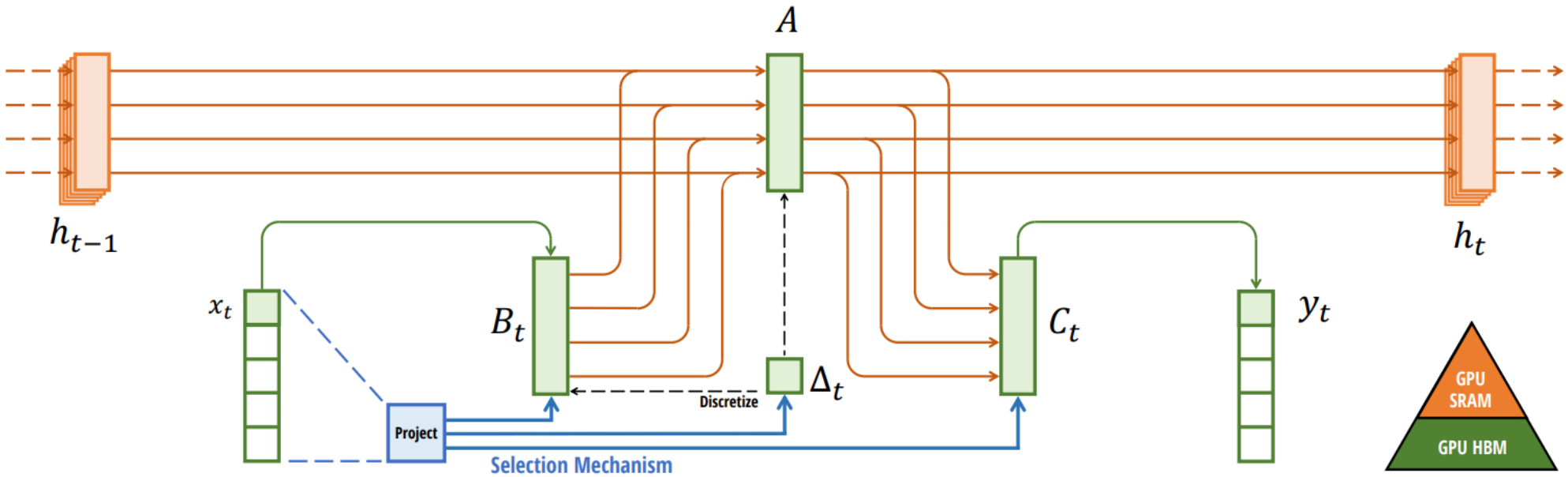
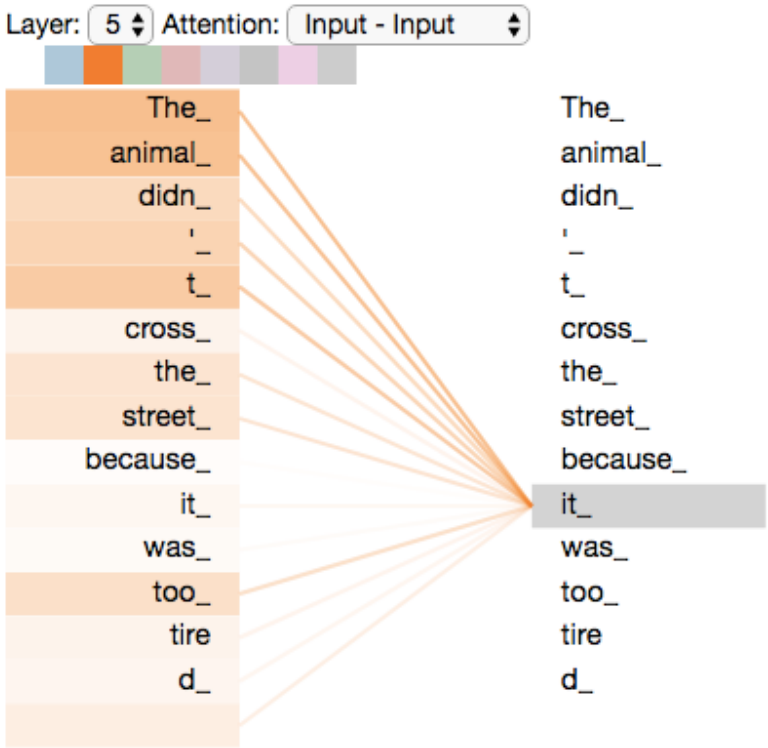


Figure 1: (**Overview**.) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ 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.

Source: Mamba (S6), Gu et al [3]

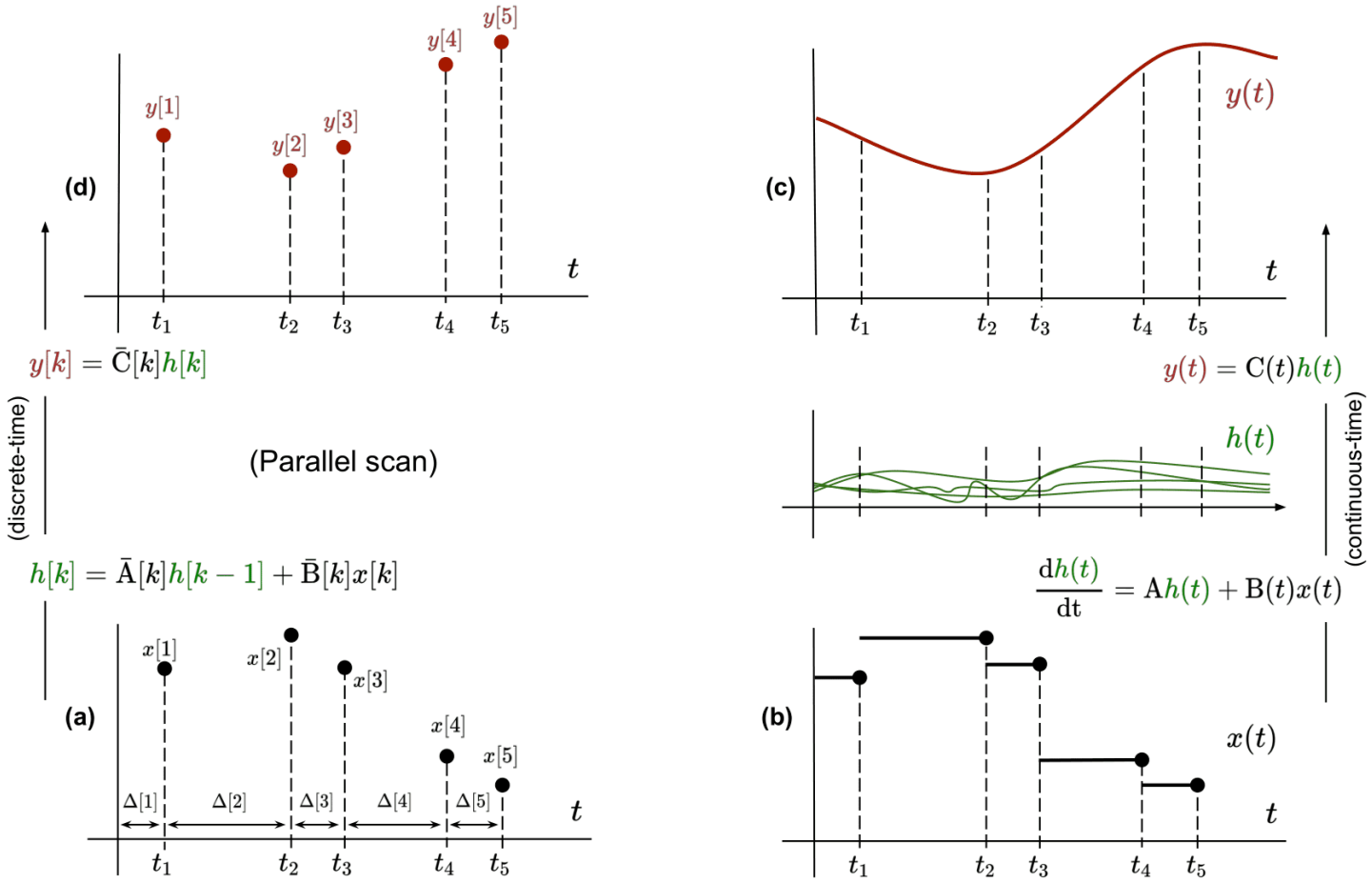
Mamba (S6): Gating or Selection Mechanism

- Cf. the attention mechanism:



Source: https://jalammar.github.io/illustrated-transformer/

Mamba (S6): Illustrated

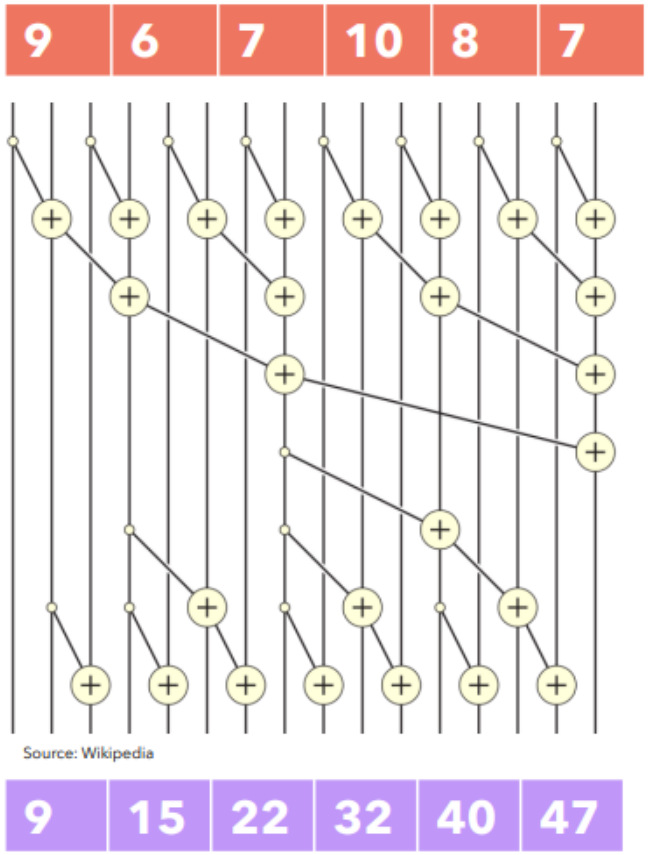


Source: Figure 2, MambaByte (Wang et al)

Mamba (S6): Associative (Parallel) Scan

- Associative scan is a parallel all-prefix sum operation.
- It takes a binary **associative** operator $\oplus$ and an ordered set (sequence) of n elements, $[a_0, \ldots, a_{n-1}]$, and returns the ordered set, $[a_0, a_0 \oplus a_1, \ldots, a_0 \oplus \cdots \oplus a_{n-1}]$.
- For example, if $\oplus$ is addition, then the scan of $[3, 1, 7, 0, 4, 1, 6, 3]$ is $[3, 4, 11, 11, 15, 16, 22, 25]$.
- This operation can be **parallelized**, as long as the operator is associative, i.e., the order of operations does not matter.
- **This can also be done for recurrences**.

Mamba (S6): Associative Scan Illustrated



Source: https://github.com/hkproj/mamba-notes

Mamba (S6): Associative Scan for Recurrence

1.4.1 First-Order Recurrences

We initially consider *first-order* recurrences of the following form

$$x_i = \begin{cases} b_0 & i = 0 \\ (x_{i-1} \otimes a_i) \oplus b_i & 0 < i < n, \end{cases} \tag{1.5}$$

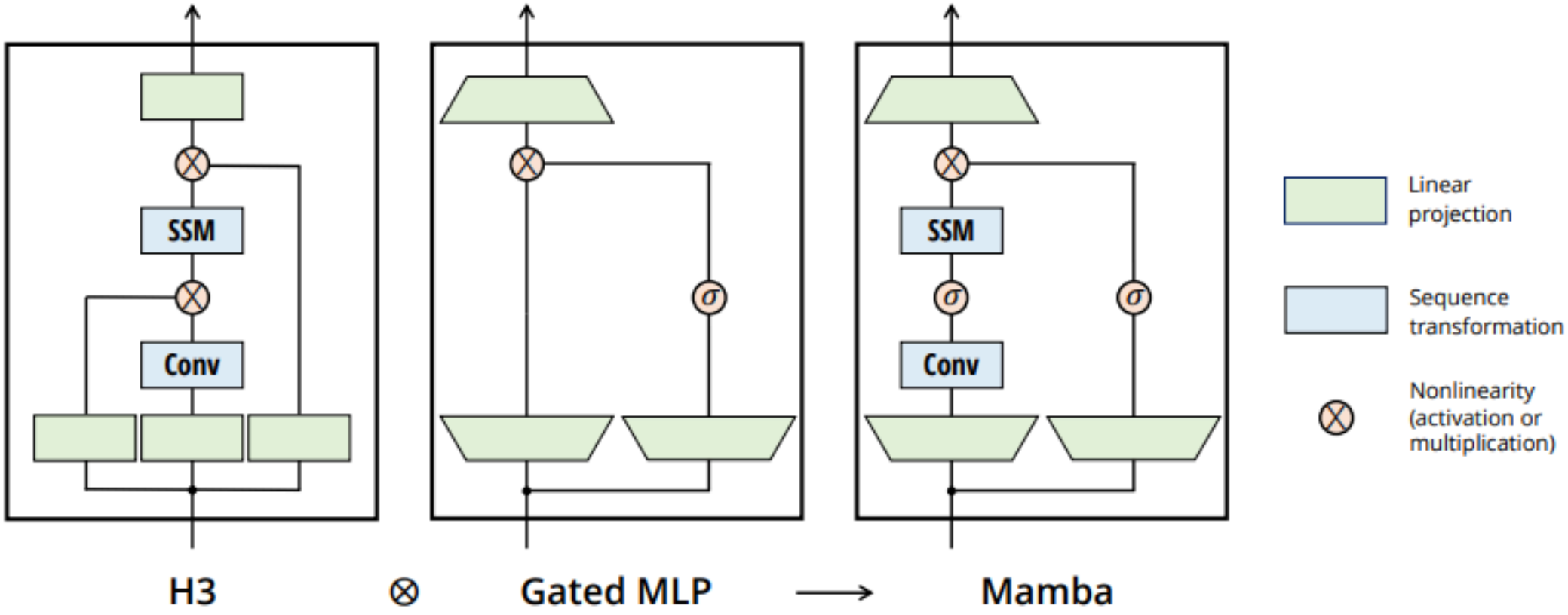
where the a_i's and b_i's are sets of n arbitrary constants (not necessarily scalars) and $\oplus$ and $\otimes$ are arbitrary binary operators that satisfy three restrictions:

1. $\oplus$ is associative (i.e. $(a \oplus b) \oplus c = a \oplus (b \oplus c)$).
2. $\otimes$ is semiassociative (i.e. there exists a binary associative operator $\odot$ such that $(a \otimes b) \otimes c = a \otimes (b \odot c)$).
3. $\otimes$ distributes over $\oplus$ (i.e. $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$).

The operator $\odot$ is called the *companion operator* of $\otimes$. If $\otimes$ is fully associative, then $\odot$ and $\otimes$ are equivalent.

Source: Prefix Sums and Their Applications (Guy E. Blelloch)

Mamba (S6): Architecture



Source: Mamba (S6), Gu et al [3]

Mamba (S6): Results

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.

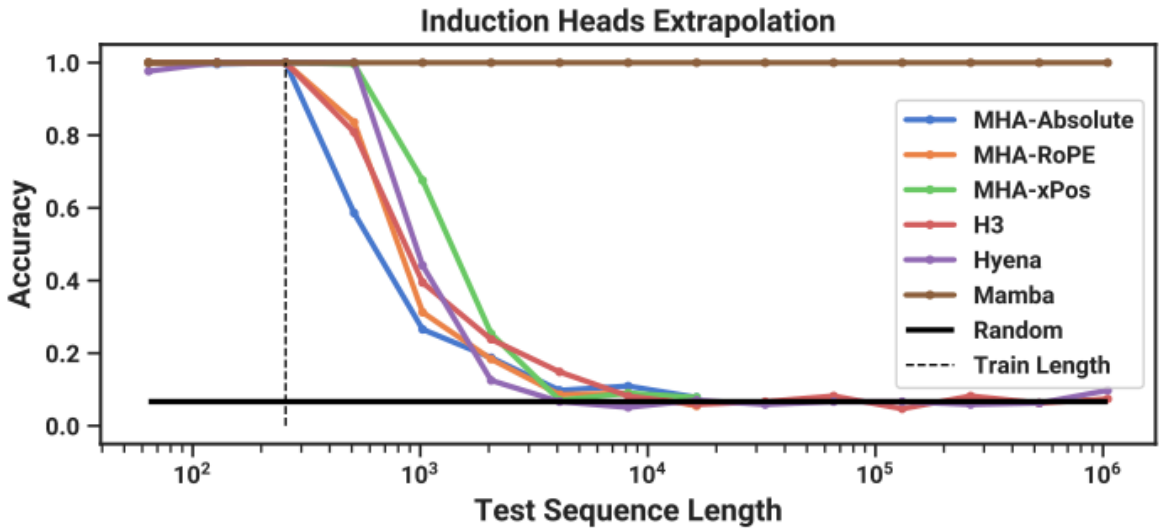


Table 2: (**Induction Heads.**) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Source: Mamba (S6), Gu et al [3]

Mamba (S6): Results

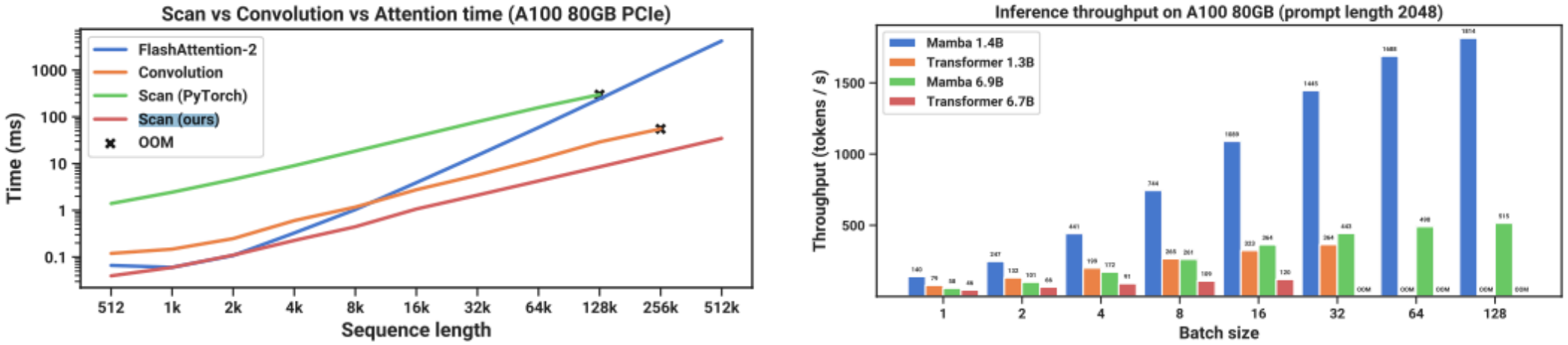


Figure 8: (**Efficiency Benchmarks**.) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

Source: Mamba (S6), Gu et al [3]

Some notes on the expressivity of SSMs

- Linear Recurrent Units (LRUs) [5] are a class of RNNs that use similar ideas to SSMs. According to the authors, these take precedence: **Linearization, Diagonalization, Stable Exponential Parameterization** and **Normalization.**
- More importantly:

> **Structured initialization is not necessary.** While Gu et al. (2022a); Gupta et al. (2022b); Smith et al. (2022) also discuss initializations for A deviating from the HiPPO structure (see §2 and §B), to the best of our knowledge we are the first to show that simple uniform initialization on a slice of the unit disk, combined with proper normalization, is able to also solve the hardest task in LRA: PathX.[11] We also show (Tb.2) that uniform initialization on the disk, which is simply the diagonalized version of Glorot initialization (Thm. 3.1), is sufficient to achieve performance close to more complex deep state-space models on the remaining LRA tasks. Our results ultimately suggest that HiPPO theory, while fundamental for the development of this field, should not be thought of as the main source of S4 success.

Source: Linear Recurrent Unit, Orvieto et al [5]

- Another recent paper (29/02/2024) has shown that chaining diagonal selective state spaces can be as expressive as dense state spaces [6].

References

1. HiPPO: Recurrent Memory with Optimal Polynomial Projections
2. S4: Efficiently Modeling Long Sequences with Structured State Spaces
3. Mamba (S6): Linear-Time Sequence Modeling with Selective State Spaces
4. Diagonal State Spaces are as Effective as Structured State Spaces
5. Resurrecting Recurrent Neural Networks for Long Sequences
6. Theoretical Foundations of Deep Selective State-Space Models

Additional Material:

1. Notes on the Mamba and the S4 model (Mamba: Linear-Time Sequence Modeling with Selective State Spaces)
2. Prefix Sums and Their Applications - Guy E. Blelloch

Thank you! Any questions?