Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Miton Times
25. 1. 2024
Radek Bartyzal

Related work

Transformers with self-attention:

- routes information densely within a context window => can model complex data
- drawbacks:
  - cannot model anything outside of a finite window
  - quadratic scaling with respect to the window length.
  - many attempts to improve this but always with worse results at scale

S4 = Structured State Space Sequence models

- efficient but not as good for text

S4 = Structured State Space Sequence models

- recurrent process with a latent state
- all transformations are linear
  - A, B, C = weight matrices
  - h_t = hidden state at time t

$$h'(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t) \qquad (1a)$$
$$y(t) = \boldsymbol{C}h(t) \qquad (1b)$$

$$h_t = \overline{\boldsymbol{A}}h_{t-1} + \overline{\boldsymbol{B}}x_t \qquad (2a)$$
$$y_t = \boldsymbol{C}h_t \qquad (2b)$$

$$\overline{\boldsymbol{K}} = (\boldsymbol{C}\overline{\boldsymbol{B}}, \boldsymbol{C}\overline{\boldsymbol{AB}}, \dots, \boldsymbol{C}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}, \dots) \qquad (3a)$$
$$y = x * \overline{\boldsymbol{K}} \qquad (3b)$$

- time-invariant = each step does the same transformation
  - => can be understood as global convolution
  - => we can calculate all time steps in parallel if we have all the inputs
- input invariant = same linear transformation for each token, no non-linearities

S4 = Structured State Space Sequence models

- sequence models
- combination of RNNs and CNNs
- linear or near-linear scaling in sequence length
- good for continuous signal data = audio
- bad for discrete and information-dense data = text
- time- and input-invariant in order to be computationally efficient
    - = each input token is transformed into latent repr. the same way
    - = simple multiply by weight matrix, no non-linearities

S4 vs Transformers

Problem with S4:

- compresses the context into the hidden state always the same way
  - all previous tokens are available only through the one hidden state
  - + it's fast = O(n)
  - - it cannot select things to pay attention to = as time goes things get lost in the compression
  - cannot solve selective copying task

Transformers:

- do not compress the context, they have representation of each token
  - - it's slow = O(n^2)
  - + it's powerful, keeps the full context
  - can solve almost any task with the finite size of context

Mamba = Selective State Space Model

- changes S4 to keep the speed = linear time with length of input
- BUT allows for selection in compressing the context

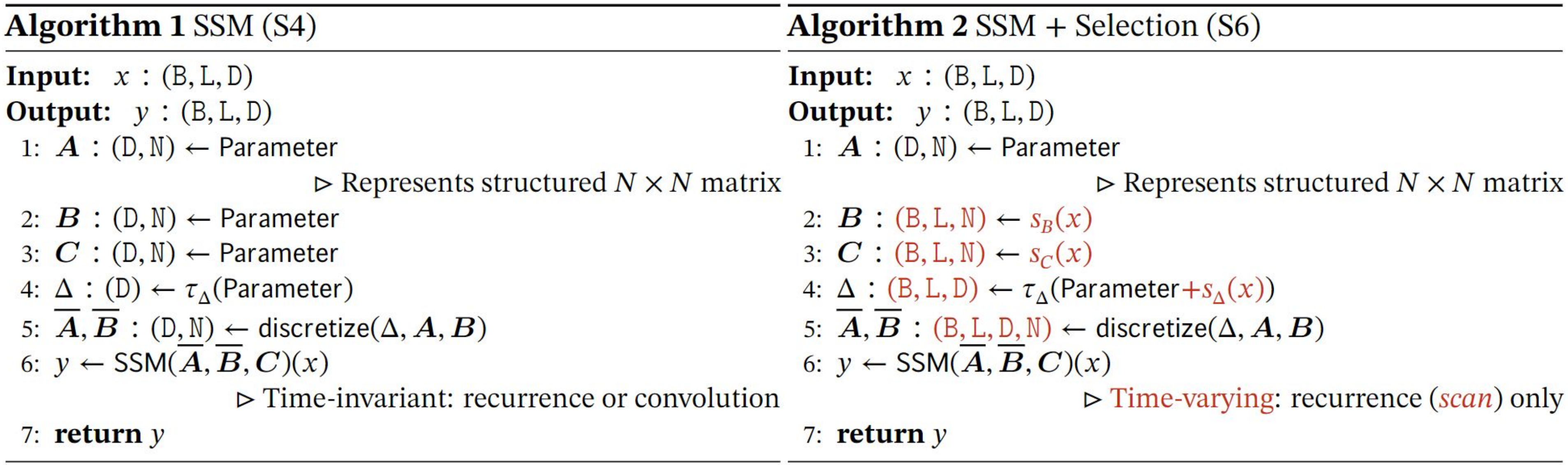
Selection = make each step input dependent

- = weight matrices are calculated based on each input token
- => it’s time variant now
- => it’s not a CNN anymore
- => has to made fast by HW specific implementation

Mamba: Selection mechanism

Selection = Make weight matrices based on input token:

- these parameters now have a length dimension L, meaning that the model has changed from time-invariant to time-varying



```
Algorithm 1 SSM (S4)
Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
                         ▷ Represents structured $N \times N$ matrix
2: $\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
3: $\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
4: $\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
                         ▷ Time-invariant: recurrence or convolution
7: return $y$
```

```
Algorithm 2 SSM + Selection (S6)
Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
1: $\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
                         ▷ 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: $\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
5: $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
6: $y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
                         ▷ Time-varying: recurrence (scan) only
7: return $y$
```

Mamba block

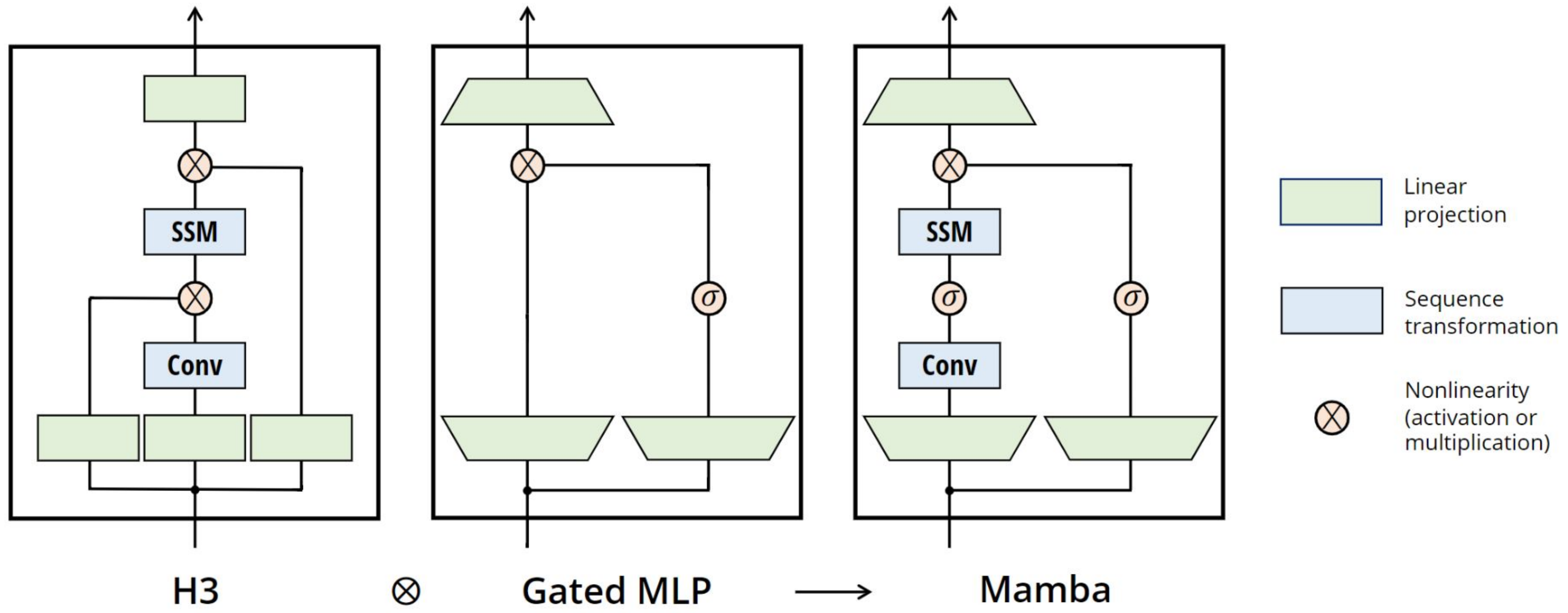


Figure 3: (**Architecture**.) Our simplified block design combines the H3 block, which is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch. For σ we use the SiLU / Swish activation (Hendrycks and Gimpel 2016; Ramachandran, Zoph, and Quoc V Le 2017).

Mamba: Selection

- added selection enables solving selective copying task

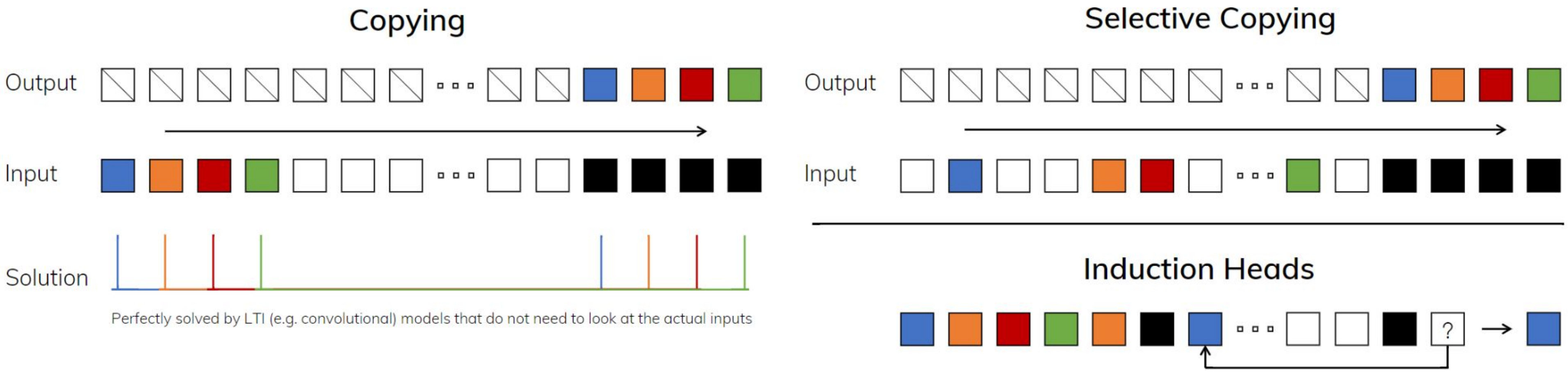
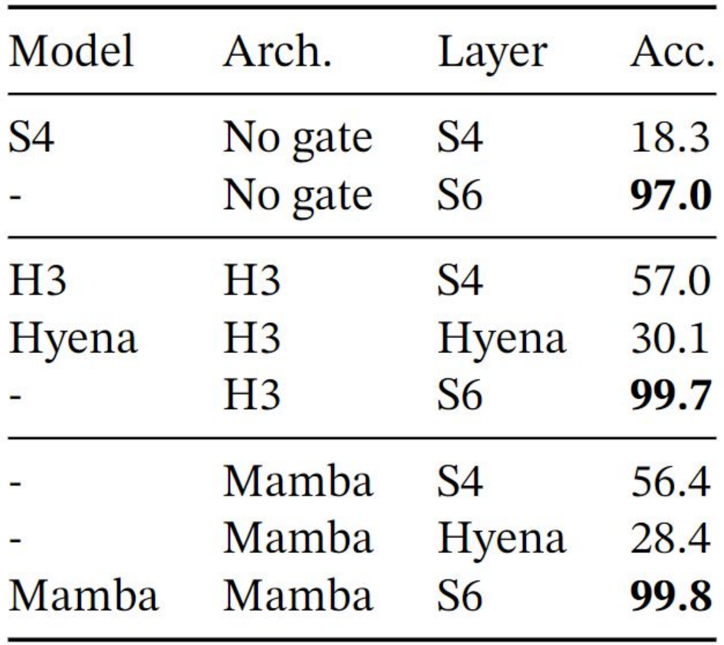

Copying
Output
Input
Solution
Perfectly solved by LTI (e.g. convolutional) models that do not need to look at the actual inputs
Selective Copying
Output
Input
Induction Heads
?

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.

Experiments: Selective Copying

- S6 = selection added

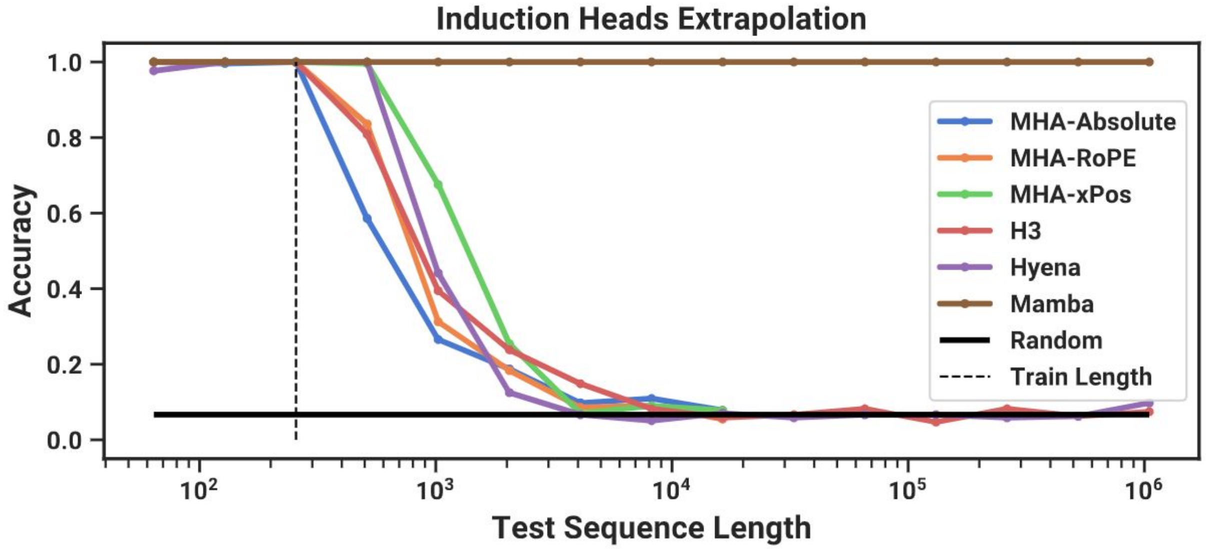


Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

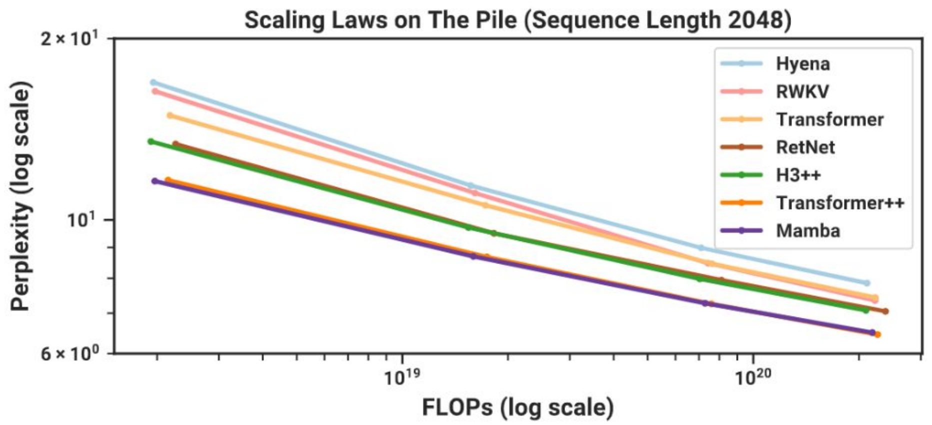
Table 1: (**Selective Copying**.) Accuracy for combinations of architectures and inner sequence layers.

Experiments: Induction heads

- if the model has seen a bigram such as “Harry Potter” in the sequence, then the next time “Harry” appears in the same sequence, the model should be able to predict “Potter” by copying from history.
- Mamba generalizes million-length sequences, or 4000×longer than it saw during training



Experiments: Language modeling



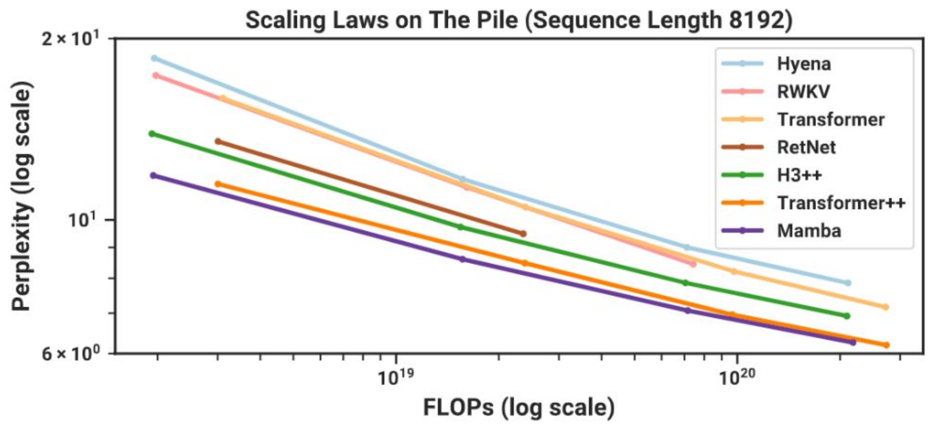


Figure 4: (**Scaling Laws**.) Models of size $\approx 125M$ to $\approx 1.3B$ parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length grows.

Experiments: Speed and MEM

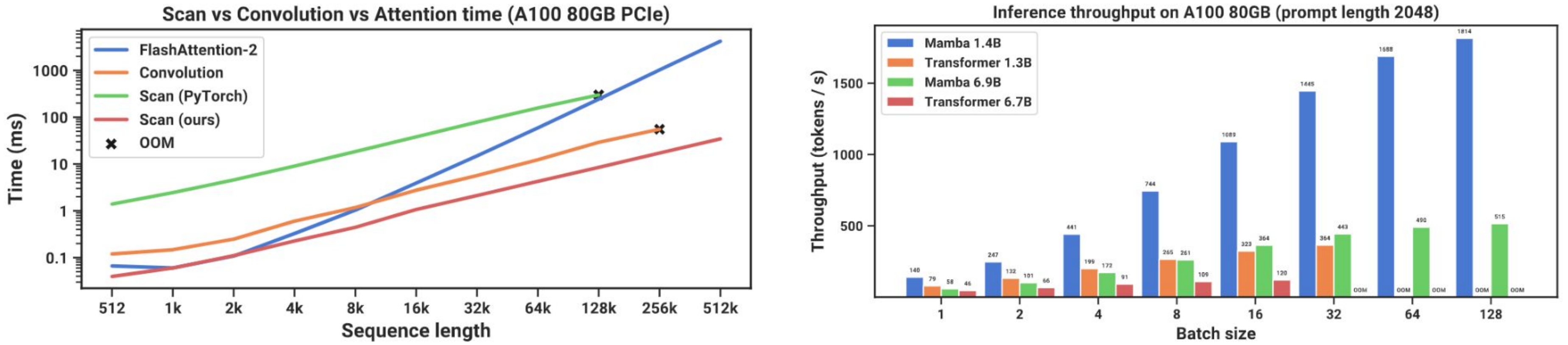


Figure 8: (**Efficiency Benchmarks**.) (*Left*) Training: our efficient scan is 40× faster than a standard implementation. (*Right*) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

Sources

paper: https://arxiv.org/abs/2312.00752