Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu and Tri Dao

Machine Learning Department, Carnegie Mellon University
Department of Computer Science, Princeton University

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering &
Artificial Intelligence

References

Mamba

- Github : https://github.com/state-spaces/mamba
- Hugging Face : https://huggingface.co/state-spaces
- Paper : https://arxiv.org/abs/2312.00752

Introduction

➢ Abstract

- Foundation models are almost universally based on the Transformer architecture and its core attention module.
- Many ***subquadratic-time architectures*** such as ***linear attention***, ***gated convolution*** **and** ***recurrent models***, and ***structured state space models (SSMs)***
  - ✓ Address Transformers' computational inefficiency on long sequences,
  - ✓ but **not performed as well as attention on important modalities such as language.**
- A key weakness of such models is their inability to perform *content-based reasoning*, and make several improvements.

1) **Letting the SSM parameters be functions of the input** addresses their weakness with discrete modalities, allowing the model to *selectively propagate or forget information along the sequence length dimension depending on the current token*.
2) Even though this change prevents the use of efficient convolutions, we design a ***hardware-aware parallel algorithm*** in *recurrent mode*.

- We integrate these ***selective SSMs*** into a *simplified end-to-end neural network architecture without attention* or *even MLP blocks (Mamba)*.
- Mamba enjoys ***fast inference*** (5× higher throughput than Transformers) and ***linear scaling in sequence length***, and its performance improves on real data up to million-length sequences.

- Mamba achieves SOTA performance across several modalities such as language, audio, and genomics. On language modeling, our **Mamba-3B** model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

Introduction

➢ *Structured* state space sequence models (SSMs)

- **SSM**s (Gu, Goel, and Ré 2022; Gu, Johnson, Goel, et al. 2021)
  - ✓ A promising class of architectures for *sequence modeling*.
  - ✓ Interpreted as a combination of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with inspiration from classical state space models (Kalman 1960).
- This class of models can be *computed very efficiently* as either a *recurrence* or *convolution*, with *linear or near-linear scaling in sequence length*.
- Many flavors of SSMs 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***.

➢ Propose a new class of *selective* state space models

- Improves on prior work on several axes to **achieve the modeling power of Transformers** while **scaling linearly in sequence length.**

[Selection Mechanism]

- Ability to efficiently *select* data in an input-dependent manner (i.e. focus on or ignore particular inputs). → A key limitation of prior models
- Building on intuition based on important synthetic tasks such as ***selective copy*** and ***induction heads***, we design a simple selection mechanism by **parameterizing the SSM parameters based on the input.**
- Filter out irrelevant information and remember relevant information indefinitely.

Introduction

➢ **Propose a new class of *selective* state space models**

[Hardware-aware Algorithm]

- Technical challenge for the computation of the model
- All prior SSMs models must be *time-* and *input-invariant* in order to be computationally efficient.

- We overcome this with ***a hardware-aware algorithm***

✓ Computes the model *recurrently with a scan* instead of *convolution*

✓ But does not materialize the expanded state in order to avoid IO access between different levels of the GPU memory hierarchy.

→ Faster than previous methods both in theory (scaling linearly in sequence length, compared to pseudo-linear for all convolution-based SSMs) and on modern hardware (up to 3× faster on A100 GPUs).

[Architecture]

- Simplify prior deep sequence model architectures by combining the design of ***prior SSM architectures*** (Dao, Fu, Saab, et al. 2023) with ***the MLP block of Transformers into a single block***, leading to **a simple and homogenous architecture design** (**Mamba**) incorporating **selective state spaces.**

- ***Selective SSMs*** **:** Fully recurrent models with key properties that make them suitable as the backbone of general foundation models operating on sequences;

  1) *High quality*: selectivity brings strong performance on dense modalities such as language and genomics.

  2) *Fast training and inference*: computation and memory scales linearly in sequence length during training, and unrolling the model autoregressively during inference requires only constant time per step since it does not require a cache of previous elements.

  3) *Long context*: the quality and efficiency together yield performance improvements on real data up to sequence length 1M.

Introduction

➢ **Propose a new class of *selective* state space models**

► Empirically validate Mamba's potential as a general sequence FM backbone, in both pretraining quality and domain specific task performance

[**Synthetics**]

- Copying and induction heads : Mamba not only solves them easily but can *extrapolate solutions indefinitely long* (>1M tokens).

[**Audio and Genomics**]

- Out-performs prior SOTA models such as *SaShiMi*, *Hyena*, and *Transformers* on modeling audio waveforms and DNA sequences

[**Language Modeling**]

- Mamba is the first *linear-time sequence model* that truly achieves Transformer-quality performance, both in pretraining perplexity and downstream evaluations.
- Mamba has 5× generation throughput compared to Transformers of similar size, and Mamba-3B's quality matches that of Transformers twice its size

2. State Space Models

➢ Structured state space sequence models (S4)

- S4 are a recent class of sequence models for deep learning that are broadly related to RNNs, and CNNs, and classical state space models.
- They are inspired by a *particular continuous system* (1) that maps a 1-dimensional function or sequence $x(t) \in \mathbb{R} \mapsto y(t) \in \mathbb{R}$ through an implicit latent state $h(t) \in \mathbb{R}^N$.
- S4 models are defined with four parameters $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, which define a sequence-to-sequence transformation in two stages.

$$h'(t) = Ah(t) + Bx(t) \quad (1a)$$
$$y(t) = Ch(t) \quad (1b)$$

Continuous system

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

Linear recurrence

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

Global convolution

2. State Space Models

➢ Discretization

- The first stage transforms the "*continuous parameters*" $(\Delta, \boldsymbol{A}, \boldsymbol{B})$ to "*discrete parameters*" $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$ through fixed formulas $\boldsymbol{A} = f_A(\Delta, \boldsymbol{A})$ and $\boldsymbol{B} = f_B(\Delta, \boldsymbol{B})$, where the pair (f_A, f_B) is called a *discretization rule*.
- A discretization rule of Zero-order hold (ZOH)

$$\overline{\boldsymbol{A}} = \exp(\Delta \boldsymbol{A}) \qquad \overline{\boldsymbol{B}} = (\Delta \boldsymbol{A})^{-1}(\exp(\Delta \boldsymbol{A}) - \boldsymbol{I}) \cdot \Delta \boldsymbol{B} \qquad (4)$$

- Discretization has deep connections to continuous-time systems which can endow them with additional properties such as resolution invariance and automatically ensuring that the model is properly normalized.
- Discretization can simply be viewed as the first step of the computation graph in the forward pass of an SSM.

➢ Computation

- After discretization; $(\Delta, \boldsymbol{A}, \boldsymbol{B}) \rightarrow (\Delta, \overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})$, the model can be computed in two ways, either as a **linear recurrence** (eq 2) or a **global convolution** (eq 3).
- The model uses the convolutional mode (eq 3) for *efficient parallelizable training* (where the whole input sequence is seen ahead of time), and switched into recurrent mode (eq 2) for *efficient autoregressive inference* (where the inputs are seen one timestep at a time).

2. State Space Models

➢ Linear Time Invariance (LTI)

- An important property of eq (1),(3) is that the model's *dynamics* are constant through time; called *linear time invariance (LTI)*,
  - ✓ Thus, $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}), (\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$ are fixed for all time-steps
- LTI is deeply connected to *recurrence* and *convolutions*.
- Think of ***LTI SSMs*** as being equivalent to any linear recurrence (eq 2a) or convolution (eq 3b), and use LTI as an umbrella term for these classes of models.
- **All structured SSMs** have been **LTI** (e.g. computed as convolutions) because of fundamental efficiency constraints.
- However, a core insight of this work is that **LTI models have fundamental limitations in modeling certain types of data**.
- **Our technical contributions involve removing the LTI constraint while overcoming the efficiency bottlenecks**.

➢ Structure and Dimensions

- Structured SSMs are so named because computing them efficiently requires imposing structure on the $\boldsymbol{A}$ matrix.
  - ✓ The most popular form of structure is *diagonal*.
- $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{N\times 1}$, $\boldsymbol{C} \in \mathbb{R}^{1\times N}$ matrices can all be represented by N numbers.
- To operate over an input sequence x of batch size B and length L with D channels, the SSM is applied independently to each channel.
- The total hidden state has dimension DN per input, and computing it over the sequence length requires $O(BLDN)$ time and memory; this is the root of the fundamental efficiency bottleneck.

2. State Space Models

➢ General State Space Models

- The term ***state space model*** has a very broad meaning which simply represents the notion of *any recurrent process with a latent state*.
- Throughout the entire paper, the term "***SSM***" to refer exclusively to *the class of structured SSMs or S4 models*.

➢ SSM Architectures

- Some of the most well-known SSM architectures; Serve as our primary baselines.
- ***Linear attention*** (Katharopoulos et al. 2020) : An approximation of *self-attention* involving *a recurrence* which can be viewed as *a degenerate linear SSM*.
- ***H3*** (Dao, Fu, et al. 2023) : *Generalize this recurrence to use S4*; it can be viewed as *an architecture with an SSM sandwiched by two gated connections* (Figure 3). H3 also inserts *a standard local convolution*, which they frame as a *shift-SSM*, before the main SSM layer.
- ***Hyena*** (Poli et al. 2023) : Use the same architecture as H3 but replaces the S4 layer with an *MLP-parameterized global convolution* (Romero et al. 2021).
- ***RetNet*** (Y. Sun et al. 2023) : Adds an *additional gate* to the architecture and uses a *simpler SSM*, allowing *an alternative parallelizable computation path*, using *a variant of multi-head attention (MHA)* instead of convolutions.
- **RWKV** (B. Peng et al. 2023) : A recent RNN designed for language modeling based on *another linear attention approximation*, *the attention-free Transformer* (S. Zhai et al. 2021). Its main "**WKV**" mechanism involves *LTI recurrences* and can be viewed as *the ratio of two SSMs*.

3. Selective State Space Models

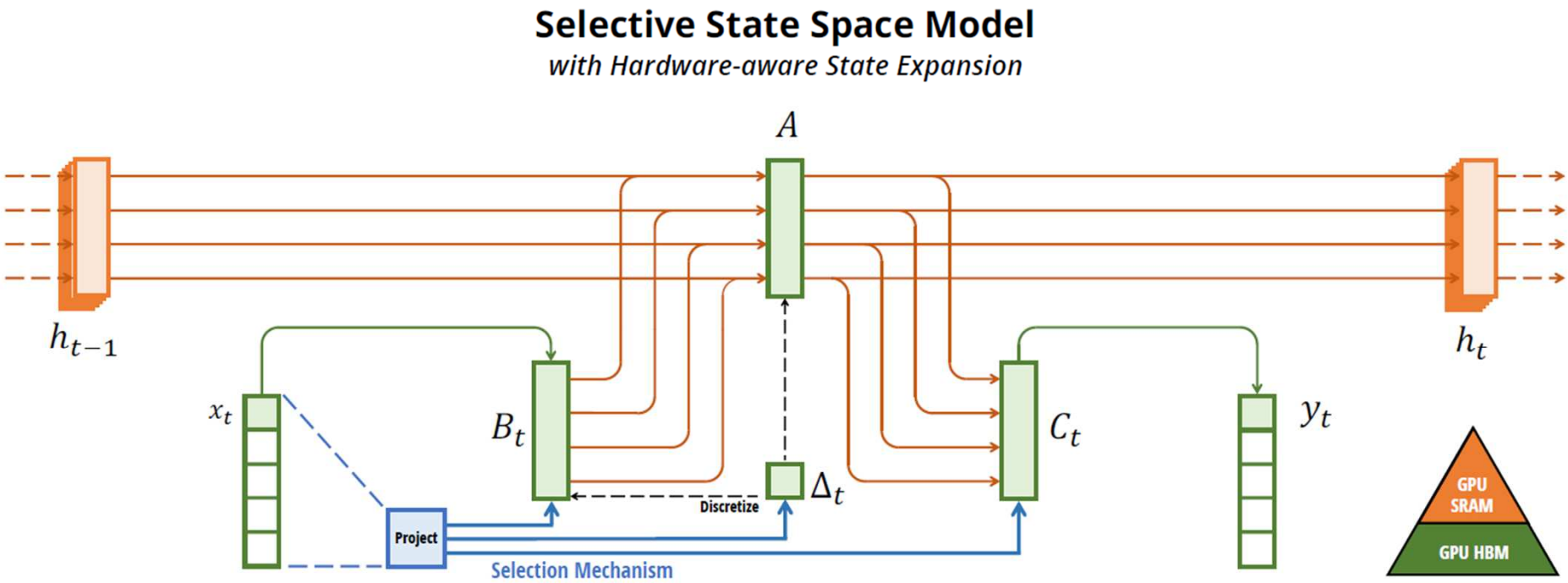


Figure 1: (Overview.) Structured SSMs independently map each channel (e.g. D=5) of an input x to output y through a higher dimensional latent state *h* (e.g. N=4).

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 (Δ,$\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$) 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.

3. Selective State Space Models

➢ **Outlines**

- Motivate *our* ***selection mechanism*** using intuition from synthetic tasks (Section 3.1),
- Explain **how to *incorporate this mechanism into state space models*** (Section 3.2).
- The resulting ***time-varying SSMs cannot use convolutions***, presenting a technical challenge of how to *compute them efficiently*.
- Overcome this with a ***hardware-aware algorithm*** that exploits the memory hierarchy on modern hardware (Section 3.3).
- Describe a **simple SSM architecture without attention or even MLP blocks** (Section 3.4).
- Finally, discuss some additional properties of selection mechanisms (Section 3.5).

3.1 Motivation: Selection as a Means of Compression

- **Fundamental problem of sequence modeling** : ***Compressing context into a smaller state*****.**
- We can view the tradeoffs of popular sequence models from this point of view.
- For example, ***attention*** is both ***effective*** and ***inefficient*** because it **explicitly does not compress context at all.**
  - ✓ **Autoregressive inference** requires explicitly **storing the entire context** (i.e. the KV cache), which directly causes the ***slow linear-time inference*** and ***quadratic-time training*** of **Transformers**.
  - ✓ **Recurrent models** are efficient because they have a finite state, implying ***constant-time inference*** *and* ***linear-time training***. However, their effectiveness is limited by how well this state has compressed the context.

3. Selective State Space Models

3.1 Motivation: Selection as a Means of Compression

- Two running examples of synthetic tasks

- **Selective Copying** task
  - Modifies the popular Copying task (Arjovsky, Shah, and Bengio 2016) by varying the position of the tokens to memorize
  - Requires ***content-aware reasoning*** to be able to memorize the relevant tokens (colored) and filter out the irrelevant ones (white)

- **Induction Heads** task
  - Well-known mechanism hypothesized to explain the majority of ***in-context learning*** abilities of LLMs (Olsson et al. 2022).
  - Requires ***context-aware reasoning*** to know when to produce the correct output in the appropriate context (*black*).

➢ These tasks reveal the failure mode of LTI models.

- *From the recurrent view*, their constant dynamics (e.g. the $(\boldsymbol{A}, \boldsymbol{B})$ transitions in (2)) cannot let them *select the correct information from their context*, or *affect the hidden state passed along the sequence in an input-dependent way*.
- *From the convolutional view*, it is known that global convolutions can solve the vanilla Copying task (Romero et al. 2021) because it *only* ***requires time-awareness***, but that they have *difficulty with the Selective Copying task* because of ***lack of content-awareness*** (Figure 2). More concretely, the spacing between inputs-to-outputs is varying and cannot be modeled by static convolution kernels

3. Selective State Space Models

3.1 Motivation: Selection as a Means of Compression

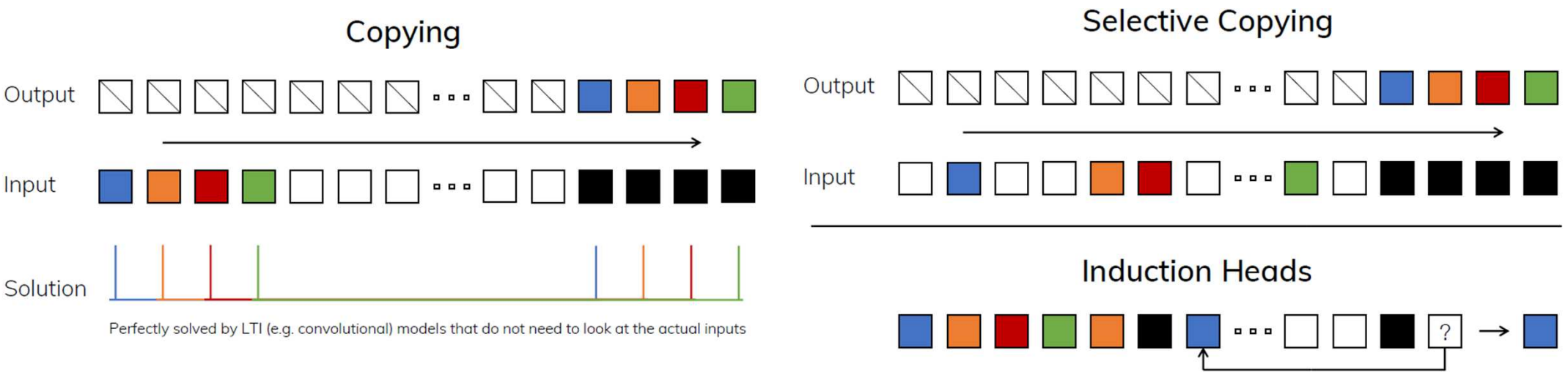


Figure 2: (Left) **The standard version of the Copying task** involves ***constant spacing between input and output elements*** and is easily solved by ***time-invariant models*** such as ***linear recurrences*** and ***global convolutions***.

(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.

3. Selective State Space Models

3.1 Motivation: Selection as a Means of Compression

- Summary

- ***Efficiency*** vs. ***effectiveness*** tradeoff of sequence models is characterized by *how well they compress their state*:
- ***Efficient models*** *must have* ***a small state***, while ***effective models*** *must have* ***a state that contains all necessary information from the context***.
- We propose that a fundamental principle for building sequence models is **selectivity**: or **the context-aware ability** to *focus on or filter out inputs into a sequential state*.
- A **selection mechanism** controls how information propagates or interacts along the sequence dimension (see Section 3.5 for more discussion).

3. Selective State Space Models

3.2 Improving SSMs with Selection

- One method of **incorporating a selection mechanism into models** is by letting their **parameters that affect interactions along the sequence** (e.g. *the recurrent dynamics of an RNN* or *the convolution kernel of a CNN*) be **input-dependent**.
- Algorithms 1 and 2 illustrates the main selection mechanism

- Main difference is making several parameters Δ, $\boldsymbol{B}$, $\boldsymbol{C}$ functions of the input, along with the associated changes to tensor shapes throughout.
- Δ, $\boldsymbol{B}$, $\boldsymbol{C}$ now have a length dimension L, meaning that the model has changed from *time-invariant* to *time-varying*. *This loses the equivalence to convolutions (3) with implications for its efficiency*, discussed next.

Algorithm 1 SSM (S4)

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

Algorithm 2 SSM + Selection (S6)

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

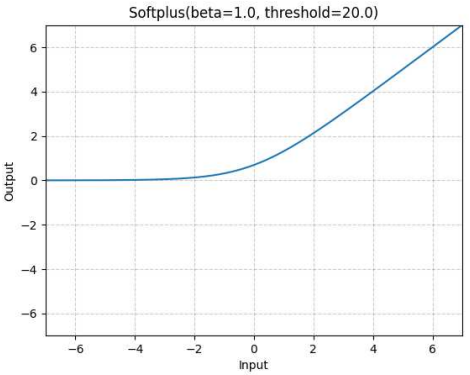
3. Selective State Space Models

3.2 Improving SSMs with Selection

- Algorithms 2 : SSM + Selection (S6)

❖ Softplus : A smooth approximation to ReLU

$$\text{Softplus}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$



- Specifically choose $s_B(x)$, $s_C(x)$, $s_\Delta(x)$

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

- Linear_d is a parameterized projection to dimension d.
- The choice of s_Δ and τ_Δ is due to a connection to RNN gating mechanisms

Algorithm 2 SSM + Selection (S6)

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

3. Selective State Space Models

3.2 Improving SSMs with Selection

```
import torch

def ssm(self, x):
    """Runs the SSM. See:
        - Algorithm 2 in Section 3.2 in the Mamba paper [1]
        - run_SSM(A, B, C, u) in The Annotated S4 [2]

    Args:
        x: shape (b, l, d_in)    (See Glossary at top for definitions of b, l, d_in, n...)

    Returns:
        output: shape (b, l, d_in)

    Official Implementation:
        mamba_inner_ref(), https://github.com/state-spaces/mamba/blob/main/mamba_ssm/ops/selective_scan_interface.py#L311
    """
    (d_in, n) = self.A_log.shape
```

```
“”” Compute ∆ A B C D, the state space parameters.
    A, D are input independent (see Mamba paper [1] Section 3.5.2 "Interpretation of A" for why A isn't selective)
    ∆, B, C are input-dependent (this is a key difference between Mamba and the linear time invariant S4, and is why Mamba is called **selective** state spaces)
    “””
    A = -torch.exp(self.A_log.float())  # shape (d_in, n)
    D = self.D.float()

    x_dbl = self.x_proj(x)  # (b, l, dt_rank + 2*n)

    (delta, B, C) = x_dbl.split(split_size=[self.args.dt_rank, n, n], dim=-1)
    # delta: (b, l, dt_rank). B, C: (b, l, n)

    delta = F.softplus(self.dt_proj(delta))  # (b, l, d_in)

    y = self.selective_scan(x, delta, A, B, C, D)
    # This is similar to run_SSM(A, B, C, u) in The Annotated S4 [2]

    return y
```

3. Selective State Space Models

3.3 Efficient Implementation of Selective SSMs

➢ Aim to make **selective SSMs** efficient on **GPUs** as well.

- **A core limitation in the usage of SSMs** is their **computational efficiency** → S4 used LTI (non-selective) models in the form of *global convolutions*.

3.3.1 Motivation of Prior Models

- Motivation and overview our approach to overcome limitations of prior methods.

- **Recurrent models** such as **SSMs** always **balance a tradeoff between *expressivity* and *speed***: models with *larger hidden state dimension* should be more *effective* but *slower*.

→ Want to **maximize hidden state dimension without paying speed and memory costs**.

- Note that ***the recurrent mode*** is more *flexible* than *the convolution mode*, since the latter (3) is derived from expanding the former (2). However, this would require computing and materializing the latent state h with shape (B, L, D, N),

- ***The more efficient convolution mode*** was introduced which could **bypass the state computation and materializes a convolution kernel (3a) of size only (B, L, D)**.

- ***Prior LTI state space models*** leverage ***the dual recurrent-convolutional forms*** to *increase the effective state dimension by a factor of* N (≈ 10 − 100), much larger than traditional RNNs, *without efficiency penalties*.

3. Selective State Space Models

3.3.2 Overview of Selective Scan: Hardware-Aware State Expansion

- The selection mechanism is designed to overcome the limitations of LTI models; at the same time, we need to revisit the computation problem of SSMs.
- We address this with three classical techniques: ***kernel fusion***, ***parallel scan***, and ***recomputation***.

- Two main observations
- The naive ***recurrent computation*** uses $O(BLDN)$ FLOPs
- The ***convolutional computation*** uses $O(BLD$ **log**$(L))$ FLOPs,
- The former has a lower constant factor. For long sequences and not-too-large state dimension N, the recurrent mode can actually use fewer FLOPs.

- The two challenges are ***the sequential nature of recurrence***, and ***the large memory usage***. To address the latter, just like the convolutional mode, we can attempt to not actually materialize the full state h.

- The main idea is to **leverage properties of GPUs to materialize the state h only in more efficient levels of the memory hierarchy**.

- Most operations (except matrix multiplication) are bounded by memory bandwidth. This includes our **scan operation**, and we use ***kernel fusion*** to reduce the amount of memory IOs, leading to a significant speedup compared to a standard implementation

①

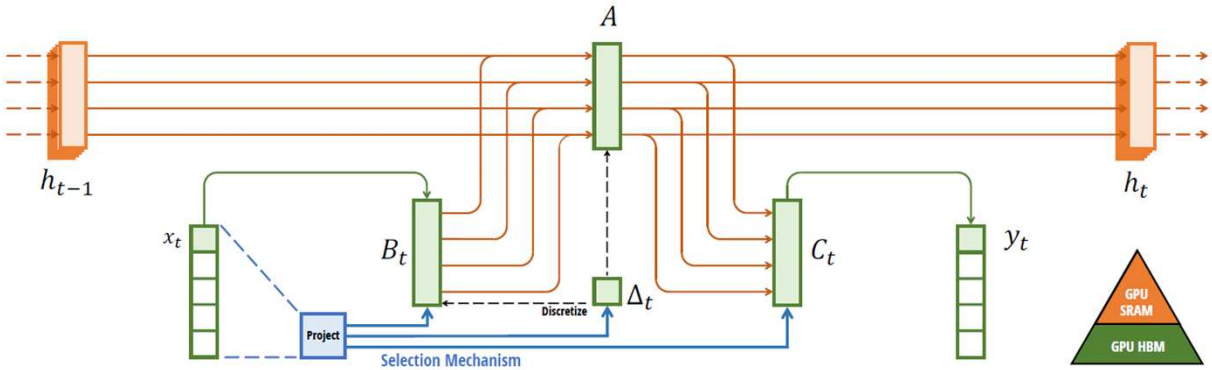
- Instead of preparing the scan input ($\boldsymbol{A}$, $\boldsymbol{B}$) of size (B, L, D, N) in slow GPU HBM (high-bandwidth memory),
- **Load SSM parameters (Δ,$\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$) directly to fast SRAM, perform the discretization and recurrence in SRAM**,
- Then **write the final outputs of size (B, L, D) back to HBM**.

3. Selective State Space Models

3.3.2 Overview of Selective Scan: Hardware-Aware State Expansion

②

- ***To avoid the sequential recurrence***, we observe that despite not being linear it can **still be parallelized with a work-efficient parallel scan algorithm** (Blelloch 1990; Martin and Cundy 2018; Smith, Warrington, and Linderman 2023).



③

- ***Must avoid saving the intermediate states***, which are necessary for backpropagation.
- We apply the classic technique of ***recomputation*** to **reduce the memory requirements**: **the intermediate states are not stored but recomputed in the backward pass when the inputs are loaded from HBM to SRAM**.
- As a result, ***the fused selective scan layer*** has **the same memory requirements** as **an optimized transformer implementation with *FlashAttention***.

- Details of the **fused kernel** and **recomputation** are in Appendix D.
- Full Selective SSM layer and algorithm is illustrated in Figure 1.

❖ Work-efficient parallel scan algorithm : Parallel Prefix Sum (Scan) with CUDA (https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda)
❖ https://blog.premai.io/s4-and-mamba/
❖ https://pli.princeton.edu/blog/2024/mamba-2-algorithms-and-systems

3. Selective State Space Models

Appendix D. Hardware-aware Algorithm For Selective SSMs

- **Without input-dependent selectivity**, **SSMs** can be efficiently implemented as a ***convolution*** (Dao, Fu, Saab, et al. 2023; Gu, Goel, and Ré 2022). **With selectivity**, **SSMs** are no-longer equivalent to convolution, but we leverage ***the parallel associative scan*.**
- While **SSM scans** are theoretically efficient (O**(**$BLDN$**) FLOPs**, **scaling linear in** L), **training foundation models with selective SSMs** requires them to be efficient on **GPUs** as well.

- We describe how we use ***kernel fusion*** and ***recomputation*** to make **SSM scan** ***fast*** and ***memory-efficient*.**
- The speed of our scan implementation is up to **7× times faster** than attention at sequence length 32K, and is as memory-efficient as the best attention implementation (***FlashAttention***).

Appendix D. Hardware-aware Algorithm For Selective SSMs

➢ Speed

- We use **kernel fusion** to **reduce the amount of memory IOs**, leading to significant speedup compared to a standard implementation.

- **Standard way to implement the scan algorithm** in Section 3.2
  - Prepare the scan input $\boldsymbol{A}, \boldsymbol{B}$ of size (B, L, D, N) in GPU HBM (high-bandwidth memory),
  - Call *a parallel associative scan* implementation to write the scan output of size (B, L, D, N) to GPU HBM,
  - then multiply that scan output with $\boldsymbol{C}$ to produce an output of size (B, L, D).

✓ However, this **requires the number of memory reads/writes on the order of $O(BLDN)$.**

➢ We can **fuse the discretization step, the scan, and the multiplication with $\boldsymbol{C}$ into one kernel**:

- **Fuse the discretization step, the scan, and the multiplication with $\boldsymbol{C}$ into one kernel;**

1) We read in $O(BLD + DN)$ bytes of memory (Δ, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$) from slow HBM to fast SRAM.
2) We **discretize** to produce $\boldsymbol{A}$, $\boldsymbol{B}$ of size (B, L, D, N) in **SRAM**.
3) We perform ***a parallel associative scan***, yielding **intermediate states of size (B, L, D, N)** in **SRAM**.
4) We **multiply and sum with $\boldsymbol{C}$, producing outputs of size (B, L, D)** and write it to HBM.

- This way **reduces IOs by a factor of $O(N)$** (the state dimension), which in practice speeds up the operation by 20-40 times (Section 4.5).

3. Selective State Space Models

Appendix D. Hardware-aware Algorithm For Selective SSMs

➢ Memory

- Describe how we use the classical technique of ***recomputation*** to **reduce the total amount of memory required to train selective SSM layers**.
- From the way we fuse **the forward pass**, we **do not save the intermediate states of size (B,L,D,N)** to avoid memory blowup. However, **these intermediate states** are **necessary for the backward pass to compute gradients**.

- We **instead *recompute* those intermediate states in the backward pass**.
- Since the inputs Δ,$\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and output gradient read from HBM to SRAM are of size $O(BLN + DN)$, and the input gradients are also of size $O(BLN + DN)$, recomputation avoids the cost of reading $O(BLND)$ elements from HBM.

➢ This means that **recomputation of the SSM states in the backward pass speeds up the computation compared to storing them and reading them from HBM**.

- We use ***recomputation*** to **optimize the memory requirement of the entire selective SSM block** (input projection, convolution, activation, scan, output projection).
- In particular, do not save intermediate activations that take a lot of memory but are ***fast to recompute*** (e.g. output of activation function or short convolution).

- As a result, ***the selective SSM layer*** has **the same memory requirement** as an **optimized Transformer** implementation with **FlashAttention**.
- **Each attention layer (FlashAttention)** stores around **12 bytes of activations per token**, an **each MLP layer** stores **around 20 bytes of activations per token**, for **a total of 32 bytes** ((assuming mixed-precision training in FP16)).
- **Each selective SSM** stores **around 16 bytes of activations per token**.

➢ Hence **2 layers of selective SSMs** have around **the same activation memory** as **an attention layer and an MLP layer**.

3. Selective State Space Models

3.4 A Simplified SSM Architecture

- As with structured SSMs, **selective SSMs** are **standalone sequence transformations** that can be flexibly incorporated into neural networks.

- The **H3 architecture** is *the basis for the most well-known SSM architectures* (Section 2), which are **generally comprised of *a block inspired by linear attention interleaved* with *an MLP (multi-layer perceptron) block***.

- We simplify this architecture by *combining these two components into one*, which is **stacked homogenously** (Figure 3).
  - ✓ Inspired by the **gated attention unit (GAU)** (Hua et al. 2022).

- This architecture involves **expanding the model dimension D by a controllable expansion factor E**.
  - ✓ For each block, most of the parameters ($3ED^2$) are in the linear projections ($2ED^2$ for input projections, ED^2 for output projection) while the inner SSM contributes less. The number of SSM parameters (projections for Δ, $\boldsymbol{B}$, $\boldsymbol{C}$, and the matrix $\boldsymbol{A}$) are much smaller in comparison.

- We **repeat this block, interleaved with standard normalization and residual connections**, to form the Mamba architecture.

- We always fix to $E = 2$ in our experiments and use **two stacks of the block to match the $12D^2$ parameters of a Transformer's interleaved MHA (multi-head attention) and MLP blocks**.

- Use the **SiLU / Swish activation** function

- Additionally use an optional normalization layer (we choose **LayerNorm** (J. L. Ba, Kiros, and Hinton 2016)), motivated by RetNet's usage of a normalization layer in a similar location (Y. Sun et al. 2023).

3. Selective State Space Models

3.4 A Simplified SSM Architecture

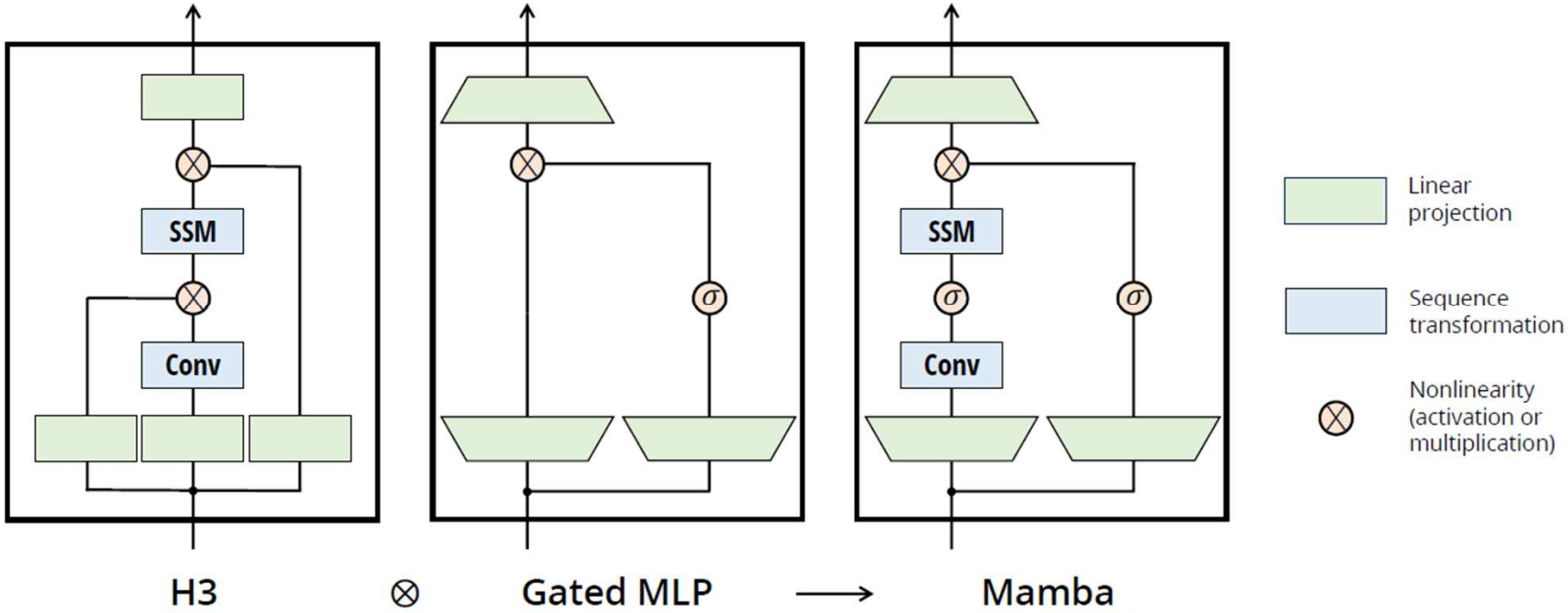


Figure 3: (**Architecture**.)

Our simplified block design combines **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).

3. Selective State Space Models

3.5 Properties of Selection Mechanisms

- The selection mechanism is a broader concept that can be applied in *different ways*, such as to more traditional **RNNs** or **CNNs**, to *different parameters* (e.g. $\boldsymbol{A}$ in Algorithm 2), or using *different transformations* $s(x)$.

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

3.5.1 Connection to Gating Mechanisms

- **The classical gating mechanism of RNNs** is **an instance of our selection mechanism for SSMs**.

- **Theorem 1** is an improvement of Gu, Johnson, Goel, et al. (2021, Lemma 3.1) generalizing to **the ZOH discretization** and **input-dependent gates** (proof in Appendix C).
- More broadly, Δ in SSMs can be seen to play a generalized role of the RNN gating mechanism.
- We adopt the view that **discretization of SSMs is the principled foundation of heuristic gating mechanisms**.

Theorem 1

- When $N = 1$, $\boldsymbol{A} = -1$, $\boldsymbol{B} = 1$, s_Δ= Linear(x), and τ_Δ = softplus, then the selective SSM recurrence (Algorithm 2) takes the form g

$$g_t = \sigma(\text{Linear}(x_t))$$
$$h_t = (1 - g_t)h_{t-1} + g_t x_t. \quad (5)$$

- Our specific choices of s_Δ, τ_Δ is from this connection.
- Note that if a given input x_t should be completely ignored, all D channels should ignore it, and so we project the input down to 1 dimension before repeating/broadcasting with Δ.

3. Selective State Space Models

Appendix C Mechanics of Selective SSMs

- Consider a selective SSM (Algorithm 2) with $N = 1$, $\boldsymbol{A} = -1$, $\boldsymbol{B} = 1$, s_Δ= Linear(x), and τ_Δ = softplus, The corresponding continuous-time SSM (1) is

$$h(t) = -h(t) + x(t)$$

called *a leaky integrator*.

- The discretization step size is

$$\begin{aligned}\Delta_t &= \tau_\Delta(\text{Parameter} + s_\Delta(x_t)) \\ &= \text{softplus}(\text{Parameter} + \text{Linear}(x_t)) \\ &= \text{softplus}(\text{Linear}(x_t))\end{aligned}$$

  ✓ the parameter can be viewed as a *learnable bias* and folded into the *linear projection*.

- Now applying the **zero-order hold** (**ZOH**) discretization formulas:

$$\begin{aligned}\overline{A}_t &= \exp(\Delta A) = \frac{1}{1+\exp(\text{Linear}(x_t))} = \sigma(-\text{Linear}(x_t)) \\ &= 1 - \sigma(\text{Linear}(x_t)) \\ \overline{B}_t &= (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B = -(\exp(\Delta A) - I) = 1 - \overline{A} \\ &= \sigma(\text{Linear}(x_t)).\end{aligned}$$

- The final discrete recurrence (2a) is

$$\begin{aligned}g_t &= \sigma(\text{Linear}(x_t)) \\ h_t &= (1 - g_t)h_{t-1} + g_t x_t\end{aligned}$$

3.5 Properties of Selection Mechanisms

3.5.2 Interpretation of Selection Mechanisms

- 3 particular mechanistic effects of **selection**

1) Variable Spacing

- Selectivity allows filtering out irrelevant noise tokens that may occur between inputs of interest.
- This is exemplified by the ***Selective Copying*** **task**, but occurs ubiquitously in common data modalities, particularly for discrete data – for example the presence of language fillers such as “um”.
- This property arises because the model can mechanistically filter out any particular input x_t, for example in the gated RNN case (Theorem 1) when $\boldsymbol{g_t \to 0}$.

2) Filtering Context

- Empirically observed that many sequence models do not improve with longer context (F. Shi et al. 2023), despite the principle that more context should lead to strictly better performance.
- *Selective models can simply reset their state at any time* to remove extraneous history, and thus their performance in principle *improves monotonicly with context length* (e.g. Section 4.3.2).

3.5 Properties of Selection Mechanisms

3.5.2 Interpretation of Selection Mechanisms

- 3 particular mechanistic effects of **selection**

3) Boundary Resetting

- In settings where *multiple independent sequences* are stitched together, **Transformers** can *keep them separate by instantiating a particular attention mask*, while **LTI** models will *bleed information between the sequences*.
- **Selective SSMs** can also *reset their state at boundaries* (e.g. $\Delta_t \to \infty$, or Theorem 1 when $g_t \to 1$). These settings may occur artificially (e.g. packing documents together to improve hardware utilization) or naturally (e.g. episode boundaries in reinforcement learning (Lu et al. 2023)).

- 3 particular mechanistic effects of **each selective parameter**

1) Interpretation of Δ

- In general, **Δ controls the balance between how much to focus or ignore the current input x_t**.
- It **generalizes RNN gates** (e.g. g_t in Theorem 1): mechanically**, a large Δ resets the state h and focuses on the current input x**, while **a small Δ persists the state and ignores the current input**.
- **SSMs eq (1)-(2)** can be interpreted as a continuous system discretized by a timestep Δ, and in this context the intuition is that large $\Delta \to \infty$ represents the system focusing on the current input for longer (thus "*selecting*" it and *forgetting* its current state) while a small $\Delta \to 0$ represents a transient input that is ignored.

3.5 Properties of Selection Mechanisms

3.5.2 Interpretation of Selection Mechanisms

- 3 particular mechanistic effects of **each selective parameter**

2) Interpretation of A

- While the A **parameter** could also be *selective*, it **ultimately affects the model only through its interaction with Δ via** $\overline{A} = \exp(\Delta A)$ (the discretization (4)).
- **Selectivity in Δ is enough to ensure selectivity in** (A, B), and is the main source of improvement.
- We hypothesize that making A selective in addition to (or instead of) Δ would have similar performance, and leave it out for simplicity.

3) Interpretation of B and C

- The most important property of selectivity is filtering out irrelevant information so that a sequence model's context can be compressed into an efficient state.
- In an SSM, **modifying** B **and** C **to be *selective*** allows **finer-grained control over whether to let an input** x_t **into the state** h_t**, or the state into the output** y_t.
- These can be interpreted as allowing the model *to* ***modulate the recurrent dynamics based on content (input) and context (hidden states) respectively***.

3.6 Additional Model Details

Real value vs. Complex value

- Most prior SSMs use **complex numbers in their state** h, which is necessary for strong performance on many tasks in *perceptual modalities* (Gu, Goel, and Ré 2022).
- However, it has been empirically observed that completely real-valued SSMs seem to work fine, and possibly even better, in some settings (Ma et al. 2023).
- We **use real values** as *the default*, which work well for all but one of our tasks; we hypothesize that **the complex-real tradeoff** is related to **the continuous-discrete spectrum in data modalities**, where **complex numbers** are helpful for **continuous modalities** (e.g. audio, video) but **not discrete** (e.g. text, DNA).

Initialization

- Most prior SSMs suggest *special initializations*, particularly in the *complex-valued case*.
- Our default initialization for *the complex case* is **S4D-Lin** *and for the real case* is **S4D-Real** (Gu, Gupta, et al. 2022), which is based on the **HIPPO theory** (Gu, Dao, et al. 2020).
- These define the n-th element of $\boldsymbol{A}$ as $-1/2 + ni$ and $-(n+1)$ respectively.
- However, we expect many initializations to work fine, particularly in the large-data and real-valued SSM regimes; some ablations are considered in Section 4.6.

3.6 Additional Model Details

Parameterization of Δ

- We defined *the selective adjustment* to Δ as $s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x))$, which was motivated by the mechanics of Δ (Section 3.5).
- It can be generalized from dimension 1 to a larger dimension R. We set this to be a *small fraction of D*, which uses a negligible number of parameters compared to the main Linear projections in the block.
- *The broadcasting operation* can instead be viewed as another Linear projection, initialized to a specific pattern of 1's and 0's; if this projection is trainable, this leads to the alternative $s_\Delta(x) = \text{Linear}_D(\text{Linear}_R(x))$, which can be viewed as a *low-rank projection*
- In our experiments, the Δ parameter (which can be viewed as a bias term) is initialized to τ_Δ^{-1} (Uniform([0.001, 0.1])), following prior work on SSMs (Gu, Johnson, Timalsina, et al. 2023).

Remark 3.1

- For brevity in our experimental results, we sometimes abbreviate *selective SSMs* as *S6* models, because they are S4 models with a *selection mechanism* and computed with a *scan*.

4 Empirical Evaluation

4.1 Synthetic Tasks

4.1.1 Selective Copying

- The ***Copying task*** is one of the most well-studied synthetic tasks for sequence modeling, originally designed to test *the memorization abilities of recurrent models*.
- **LTI SSMs** (***linear recurrences*** and ***global convolutions***) can easily solve this task by *only keeping track of time instead of reasoning about the data*; for example, by constructing a convolution kernel of exactly the right length (Figure 2).
- The **Selective Copying task** prevents this shortcut by randomizing the spacing between tokens.
- Table 1 confirms that *gated architectures* such as **H3** and **Mamba** only partially improve performance, while the *selection mechanism* (**modifying S4 to S6**) easily solves this task, particularly when combined with these more powerful architectures
- Many previous works argue that adding ***architecture gating*** (*multiplicative interactions*) can endow models with "***data-dependence***" and solve related tasks (Dao, Fu, Saab, et al. 2023; Poli et al. 2023). However, we find this explanation insufficient intuitively because such gating does not interact along the sequence axis, and cannot affect the spacing between tokens. In particular *architecture gating* is *not an instance of a selection mechanism* (Appendix A).

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.

4 Empirical Evaluation

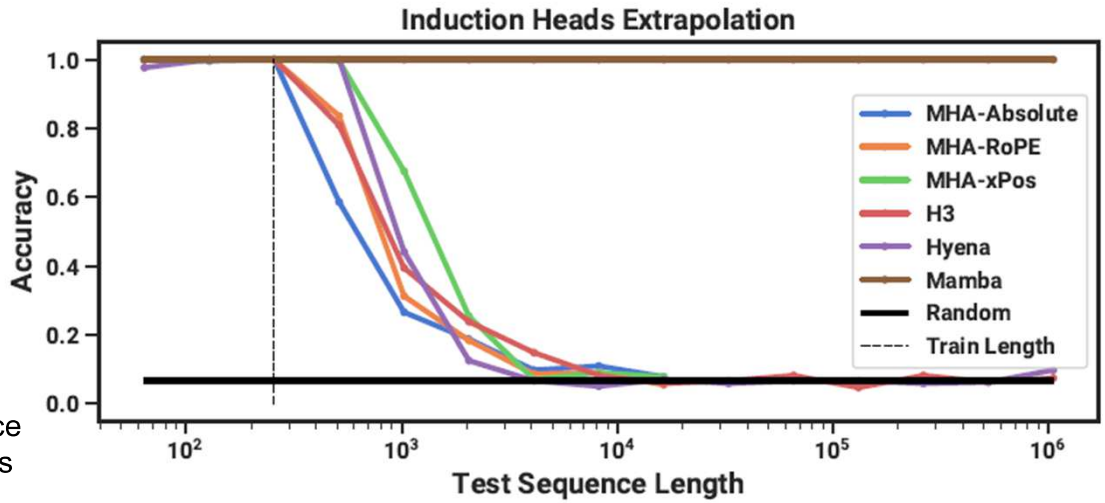
4.1 Synthetic Tasks

4.1.2 Induction Heads

- **Induction heads** (Olsson et al. 2022) is a simple task from the mechanistic interpretability lens (Elhage et al. 2021) that is *surprisingly predictive of the in-context learning ability of LLMs*. It requires models to perform *associative recall and copy*:
- Ex : 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.
- [**Dataset**] : We train a *2-layer model* on *the induction heads task* at *sequence length 256*, with a vocab size of 16, which is comparable to prior work on this task but with longer sequences. We additionally investigate *generalization* and *extrapolation* abilities by evaluating on a range of sequence lengths from 26 = 64 up to 220 = 1048576 at test time.

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.

- [**Models**] : Following established work on induction heads, we use *2 layer models*, which allows attention to mechanistically solve the induction heads task. We test both *multi-head attention* (8 heads, with various positional encodings) and *SSM* variants. We use a model dimension D of 64 for Mamba and 128 for the other models.
- [**Result**] Table 2 shows that *Mamba*—or more precisely, its *selective SSM layer*—has the ability to solve the task perfectly because of its ability to selectively remember the relevant token while ignoring everything else in between. **It generalizes perfectly to million-length sequences, or 4000× longer than it saw during training**, while no other method goes beyond 2×.



4 Empirical Evaluation

4.2 Language Modeling

- Evaluate the Mamba architecture on *standard autoregressive language modeling* against other architectures, on both *pretraining metrics* (**perplexity**) and *zero-shot evaluations*.
- We set *the model sizes (depth and width)* to *mirror GPT3* specifications. We use the **Pile dataset** (L. Gao, Biderman, et al. 2020), and follow the training recipe described in Brown et al. (2020).
- All training details are in Appendix E.2.

4.2.1 Scaling Laws

- For baselines, we compare against the standard Transformer architecture (**GPT3** architecture), as well as the strongest Transformer recipe we know of (here referred to as Transformer++), based on the **PaLM** and **LLaMa** architectures (e.g. rotary embedding, SwiGLU MLP, RMSNorm instead of LayerNorm, no linear bias, and higher learning rates).
- We also compare against other recent subquadratic architectures (Figure 4). All model details are in Appendix E.2.

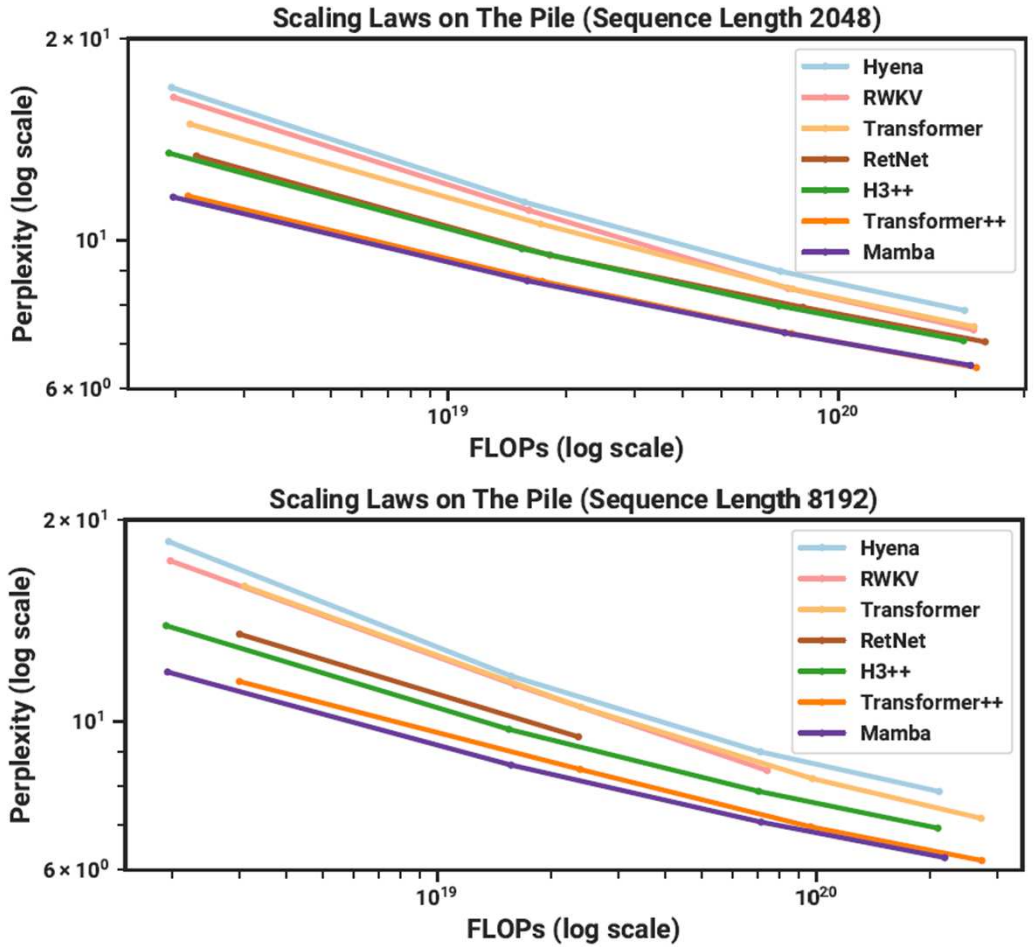


Figure 4: (**Scaling Laws**.) Models of size ≈ 125M to ≈ 1.3B parameters, trained on the **Pile**. **Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length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

4 Empirical Evaluation

4.2 Language Modeling

4.2.2 Downstream Evaluations

- Table 3 shows the performance of Mamba on a range of popular downstream zero-shot evaluation tasks.
- We compare against the most well-known open source models at these sizes, most importantly **Pythia** (Biderman et al. 2023) and **RWKV** (B. Peng et al. 2023) which were trained with the same tokenizer, dataset, and training length (300B tokens) as our models. (Note that Mamba and Pythia are trained with context length 2048, while RWKV was trained with context length 1024.)

Table 3 (**Zero-shot Evaluations**.) Best results for each size in bold. We compare against open source LMs with various tokenizers, trained for up to 300B tokens. Pile refers to the validation split, comparing only against models trained on the same dataset and tokenizer (GPT-NeoX-20B). For each model size, Mamba is best-in-class on every single evaluation result, and generally matches baselines at twice the model size.

Model	Token.	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
Hybrid H3-130M	GPT2	—	89.48	25.77	31.7	64.2	44.4	24.2	50.6	40.1
Pythia-160M	NeoX	29.64	38.10	33.0	30.2	61.4	43.2	24.1	**51.9**	40.6
Mamba-130M	NeoX	**10.56**	**16.07**	**44.3**	**35.3**	**64.5**	**48.0**	**24.3**	**51.9**	**44.7**
Hybrid H3-360M	GPT2	—	12.58	48.0	41.5	68.1	51.4	24.7	54.1	48.0
Pythia-410M	NeoX	9.95	10.84	51.4	40.6	66.9	52.1	24.6	53.8	48.2
Mamba-370M	NeoX	**8.28**	**8.14**	**55.6**	**46.5**	**69.5**	**55.1**	**28.0**	**55.3**	**50.0**
Pythia-1B	NeoX	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	51.9
Mamba-790M	NeoX	**7.33**	**6.02**	**62.7**	**55.1**	**72.1**	**61.2**	**29.5**	**56.1**	**57.1**
GPT-Neo 1.3B	GPT2	—	7.50	57.2	48.9	71.1	56.2	25.9	54.9	52.4
Hybrid H3-1.3B	GPT2	—	11.25	49.6	52.6	71.3	59.2	28.1	56.9	53.0
OPT-1.3B	OPT	—	6.64	58.0	53.7	72.4	56.7	29.6	59.5	55.0
Pythia-1.4B	NeoX	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
RWKV-1.5B	NeoX	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	54.3
Mamba-1.4B	NeoX	**6.80**	**5.04**	**64.9**	**59.1**	**74.2**	**65.5**	**32.8**	**61.5**	**59.7**
GPT-Neo 2.7B	GPT2	—	5.63	62.2	55.8	72.1	61.1	30.2	57.6	56.5
Hybrid H3-2.7B	GPT2	—	7.92	55.7	59.7	73.3	65.6	32.3	61.4	58.0
OPT-2.7B	OPT	—	5.12	63.6	60.6	74.8	60.8	31.3	61.0	58.7
Pythia-2.8B	NeoX	6.73	5.04	64.7	59.3	74.0	64.1	32.9	59.7	59.1
RWKV-3B	NeoX	7.00	5.24	63.9	59.6	73.7	67.8	33.1	59.6	59.6
Mamba-2.8B	NeoX	**6.22**	**4.23**	**69.2**	**66.1**	**75.2**	**69.7**	**36.3**	**63.5**	**63.3**
GPT-J-6B	GPT2	–	4.10	68.3	66.3	75.4	67.0	36.6	64.1	63.0
OPT-6.7B	OPT	–	4.25	67.7	67.2	76.3	65.6	34.9	65.5	62.9
Pythia-6.9B	NeoX	6.51	4.45	67.1	64.0	75.2	67.3	35.5	61.3	61.7
RWKV-7.4B	NeoX	6.31	4.38	67.2	65.5	76.1	67.8	37.5	61.0	62.5

4 Empirical Evaluation

4.3 DNA Modeling

- 논문 참조

4.4 Audio Modeling and Generation

- For the ***audio waveform modality***, we compare primarily to the **SaShiMi** architecture and training protocols (Goel et al. 2022). This model comprises:
    1. A U-Net backbone with two stages of pooling by a factor p that doubles the model dimension D per stage,
    2. Alternating S4 and MLP blocks in each stage.
- We consider replacing the **S4+MLP blocks** with **Mamba blocks**. Experiment details are in Appendix E.4.

4.4.1 Long-Context Autoregressive Pretraining

4.4.2 Autoregressive Speech Generation

- 논문 참조

4.5 Speed and Memory Benchmarks

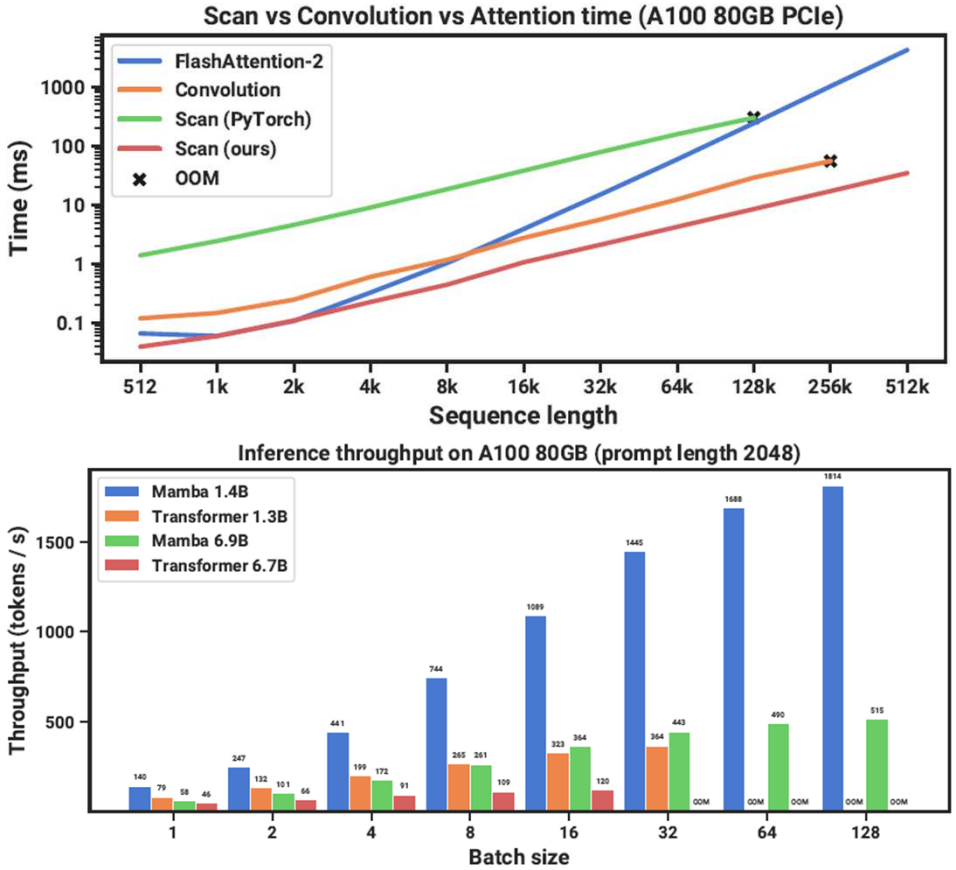


Figure 8: (**Efficiency Benchmarks**.) (Top) Training: our efficient scan is 40× faster than a standard implementation. (Bottom) Inference: as a recurrent model, Mamba can achieve 5× higher throughput than Transformers.

4 Empirical Evaluation

4.6 Model Ablations

- We perform a series of detailed ablations on components of our model, focusing on the setting of language modeling with size ≈ 350M models at Chinchilla token counts (same setting as Figure 4).

4.6.1 Architecture

- Table 6 investigates the effects of the architecture (block) and its inner SSM layer (Figure 3).
- We find that;

✓ Among previous *non-selective (LTI) SSMs*, which are equivalent to global convolutions, performance is very similar.

✓ Replacing the *complex-valued S4 variant* from previous work with a *real-valued one* does *not affect performance much*, suggesting that (at least for LM) real-valued SSMs may be a better choice when accounting for hardware efficiency.

✓ Replacing any of these with *a selective SSM (S6)* significantly improves performance, validating the motivation of Section 3.

✓ The Mamba architecture performs similarly to the H3 architecture (and seems slightly better when using a selective layer).

Table 6: (Ablations: **Architecture and SSM layer**.) The **Mamba** block performs *similarly* to **H3** while being simpler. In the inner layer, there is little difference among different parameterizations of LTI models, while selective SSMs (S6) provide a large improvement. More specifically, the S4 (real) variant is S4D-Real and the S4 (complex) variant is S4D-Lin.

Model	Arch.	SSM Layer	Perplexity
Hyena	H3	Hyena	10.24
H3	H3	S4 (complex)	10.30
-	H3	S4 (real)	10.34
-	H3	S6	**8.95**

Model	Arch.	SSM Layer	Perplexity
-	Mamba	Hyena	10.75
-	Mamba	S4 (complex)	10.54
-	Mamba	S4 (real)	10.56
Mamba	Mamba	S6	**8.69**

4.6 Model Ablations

4.6.2 Selective SSM

Table 7: (Ablations: **Selective parameters**.) Δ is the most important parameter (Theorem 1), but using multiple selective parameters together synergizes.

Selective Δ	Selective B	Selective C	Perplexity
✗	✗	✗	10.93
✗	✓	✗	10.15
✗	✗	✓	9.98
✓	✗	✗	9.81
✓	✓	✓	8.71

Table 8: (Ablations: **Parameterization** of $\boldsymbol{A}$.) The more standard initializations based on S4D-Lin (Gu, Gupta, et al. 2022) perform worse than S4D-Real or a random initialization, when the SSM is selective.

A_n Initialization	Field	Perplexity
$A_n = -\frac{1}{2} + ni$	Complex	9.16
$A_n = -1/2$	Real	8.85
$A_n = -(n+1)$	Real	8.71
$A_n \sim \exp(\mathcal{N}(0,1))$	Real	8.71

4 Empirical Evaluation

4.6 Model Ablations

4.6.2 Selective SSM

Table 9: (Ablations: **Expressivity** of Δ.) The selection mechanism of Δ constructs it with a projection of the input. Projecting it even to dim. 1 provides a large increase in performance; increasing it further provides further improvements at the cost of a modest increase in parameters. State size fixed to $N = 16$.

Size of Δ proj.	Params (M)	Perplexity
-	358.9	9.12
1	359.1	8.97
2	359.3	8.97
4	359.7	8.91
8	360.5	8.83
16	362.1	8.84
32	365.2	8.80
64	371.5	8.71

Table 10: (Ablations: **SSM state dimension**.) (Top) Constant $\boldsymbol{B}$ and $\boldsymbol{C}$ (Bottom) Selective $\boldsymbol{B}$ and $\boldsymbol{C}$. Increasing the SSM state dimension N, which can be viewed as an expansion factor on the dimension of the recurrent state, can significantly improve performance for a negligible cost in parameters/FLOPs, but only when $\boldsymbol{B}$ and $\boldsymbol{C}$ are also selective. Size of Δ projection fixed to 64.

State dimension N	Params (M)	Perplexity
1	367.1	9.88
2	367.4	9.86
4	368.0	9.82
8	369.1	9.82
16	371.5	9.81
1	367.1	9.73
2	367.4	9.40
4	368.0	9.09
8	369.1	8.84
16	371.5	8.71

Why Mamba was rejected on ICLR2024?

https://medium.com/@jelkhoury880/why-mamba-was-rejected-9b2f05f2141c
Joe El Khoury - GenAI Engineer

최근 ICLR는 2024년 컨퍼런스에 대한 최종 결정을 발표하면서 특정 제출물인 Mamba 모델에 큰 관심을 모았음.

처음에는 언어 모델링 작업에서 잘 알려진 Transformer 아키텍처의 주요 경쟁자로 여겨졌던 이 모델은 리뷰어들로부터 8 – 8 – 6 – 3의 점수를 받으며 유망한 것으로 평가되었지만 결국 최종적으로 채택되지 못했음.

Why Was Mamba Rejected?

특히 컨텍스트 길이에 따라 선형적으로 확장할 수 있는 선택적 상태 공간 모델이라는 혁신적인 접근 방식과 특정 측면에서 트랜스포머보다 성능이 뛰어날 수 있다는 점을 고려하면 Mamba의 Reject는 의문을 불러일으킴.

하지만 심사 위원들의 피드백을 자세히 살펴보면, 우려는 주로 평가 방법론에 대한 것이었음이 분명해짐.

Two critical issues led to Mamba's rejection:

1. Missing LRA Results:

Long Sequence 모델을 평가하는 기준인 LRA(Long Range Arena) 벤치마크 결과가 누락됨. LRA는 유사한 연구에서 통상적인 벤치마크로 사용되어 왔기 때문에 누락된 것은 주목할 만한 사항임.

2. Questioning Perplexity as an Evaluation Metric

난해성(Perplexity)을 주요 지표로 사용하는 것에 대한 의문이 제기되었음. 비평가들은 난해성 점수가 낮다고 해서 반드시 실제 NLP 애플리케이션의 모델링 능력이 더 뛰어난 것은 아니며, 보다 포괄적인 평가 방법이 필요하다고 주장함.

논문의 강점에도 불구하고 포괄적인 벤치마크가 부족하고 Perplexity에 의존하고 있다는 점이 상당한 것으로 간주됨. 검토 과정은 추가 실험을 통해 이러한 문제를 해결하는 것이 향후 제출에 도움이 될 수 있음을 시사하는 기각 권고로 마무리됨.

The Future of Rejected Masterpieces

Mamba와 Word2vec의 경험은 학술 연구의 중요한 측면, 즉 학회에서의 거절이 연구의 가치나 잠재적 영향력을 정의하지 않는다는 점을 강조함.

나중에 NeurIPS 2023 테스트 오브 타임 상을 수상한 Word2vec의 사례에서 볼 수 있듯이, 획기적인 아이디어는 초기 좌절을 뛰어넘어 해당 분야에 크게 기여할 수 있음.

또한, 최근 인기 있는 오픈소스 멀티모달 대형 모델인 CogVLM이 ICLR에 의해 거부된 사례는 연구의 여정이 도전과 학습 기회로 가득 차 있다는 것을 더욱 강조함.

Mamba, CogVLM 및 이와 유사한 프로젝트의 연구자에게는 거부가 더 많은 개선과 궁극적인 인정을 위한 디딤돌이 될 수 있음.

A Reflection on the Word2vec Rejection

Mamba의 이야기는 또 다른 획기적인 작품의 여정과 유사합니다:

Word2vec. 2013년 ICLR의 첫 번째 컨퍼런스에서 처음 거절당했던 Word2vec은 이후 NLP의 초석이 되어 이 분야에서 예측할 수 없는 혁신적인 연구의 길을 보여주었음.

주로 제출자의 발표 및 피드백에 대한 응답과 관련된 거절 사유는 동료 심사 과정의 복잡성을 강조함.