Selective State Spaces
Bridging the Gap Between RNNs and Transformers

Clement Marie • Ying JIN
Course: Deep Learning Project

1. Motivation & Goal

The Problem: While Transformers dominate deep learning, their quadratic complexity ($O(N^2)$) limits scalability. Mamba (SSM) offers linear-time ($O(N)$) inference, but official implementations are opaque (CUDA-heavy) and strictly causal (unidirectional).

Our Goal:

1. Provide a transparent **Pure PyTorch** implementation of Selective Scan (S6).
2. Extend Mamba with **Bi-directional Inductive Bias** for Vision tasks.

2. Method: Mamba Architecture

Mamba replaces attention with an input-dependent SSM. Key components: **Learned Discretization** Δ, **Causal Conv1d**, and **Gating**.

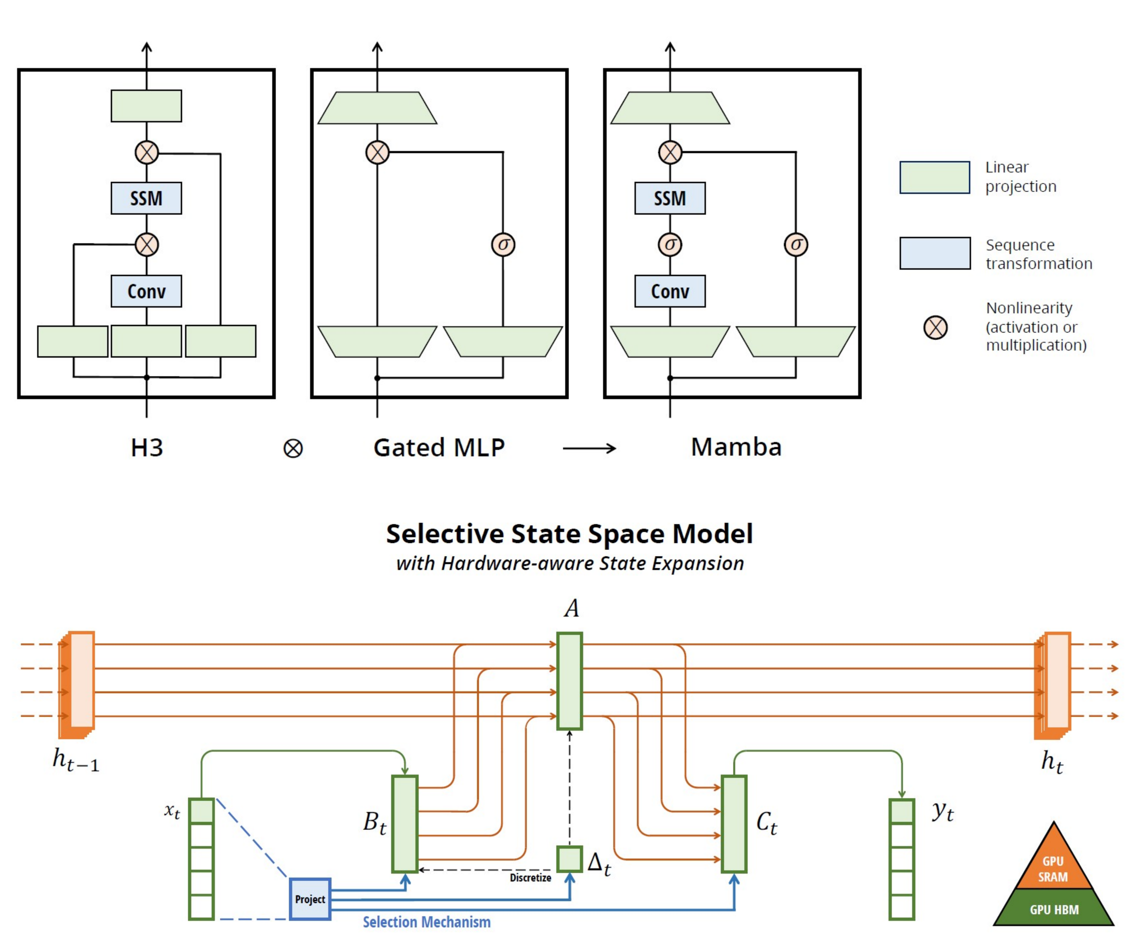

Figure 1: (Top) Gated block with SiLU and residual; (Bottom) Selection mechanism where learned Δ acts as a gate.

From Continuous to Discrete: Discretizing the continuous ODE $h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t)$ with a learned time-step Δ:

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

where $\overline{\mathbf{A}}_t = \exp(\boldsymbol{\Delta}_t \mathbf{A})$, $\quad \overline{\mathbf{B}}_t = (\boldsymbol{\Delta}_t \mathbf{A})^{-1}(\overline{\mathbf{A}}_t - \mathbf{I},$

Innovation: Δ_t depends on input x_t, allowing the model to selectively "remember".

3. Algorithm: Selective Scan (S6)

The input-dependent recurrence is computed via a parallel scan.

Algorithm 1 SSM + Selection (S6)
Require: Input $x : (B, L, D)$
Ensure: Output $y : (B, L, D)$
$A : (D, N) \leftarrow$ Parameter ▷ Fixed structured matrix
// 1. Selection: Projects input to parameters
$B : (B, L, N) \leftarrow \text{Linear}_B(x)$
$C : (B, L, N) \leftarrow \text{Linear}_C(x)$
$\Delta : (B, L, D) \leftarrow \text{Softplus}(\text{Parameter} + \text{Linear}_\Delta(x))$
// 2. Discretization
$\overline{A}, \overline{B} \leftarrow \text{discretize}(\Delta, A, B)$
// 3. Selective Scan (The parallel recurrence)
$y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$
return y

4. Implementation Details

- **Framework:** Pure PyTorch (No custom CUDA kernels).
- **Conv1d:** Kernel size 4, helps with local context.
- **Normalization:** RMSNorm for stability.
- **Optimization:** Weight tying between embeddings and output head.

The code can be found at : https://github.com/clmrie/Mamba-S6-From-Scratch.

5. Vision Extension: Bi-Directional

Why Bi-directional? Standard Mamba lose context from "future" pixels, which is suboptimal for images. Bi-directional scanning captures global spatial context. We implement a **Bi-Directional Vision Mamba**:

1. Flatten image into patch tokens.
2. Scan Forward ($\rightarrow$) and Backward ($\leftarrow$).
3. Fuse branches (Average/Concat).

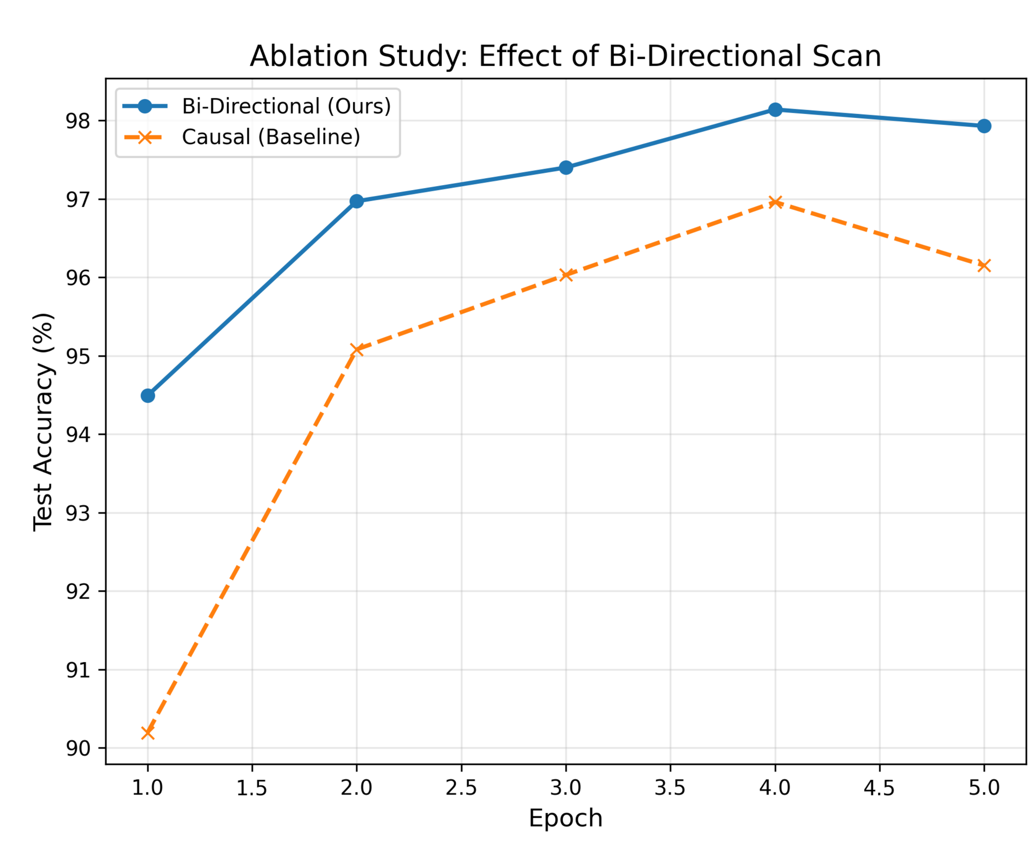

Figure 2: Causal vs. Bi-Directional Performance.

6. Experimental Setup

- MNIST: Vision Mamba, causal ablation, vanilla RNN baseline.
- CIFAR-10: patch-based classification.
- TinyShakespeare: Character-level LM.

Dataset	Key training settings
MNIST (Vision)	d_model=64, layers=2, batch=64, lr=1e-3, epochs=5
MNIST (Causal)	d_model=64, layers=2, batch=64, lr=3e-4, epochs=5
MNIST (RNN)	input=16, hidden=64, layers=2, batch=64, lr=1e-3, epochs=5
CIFAR-10	d_model=128, layers=4, batch=64, lr=1e-3, epochs=5
TinyShakespeare	d_model=128, layers=4, block=128, batch=32, lr=1e-3, epochs=10

Table 1: Training settings summary

7. Results: Mamba vs RNN

Vision Mamba converges significantly faster than vanilla RNNs and achieves higher final accuracy.

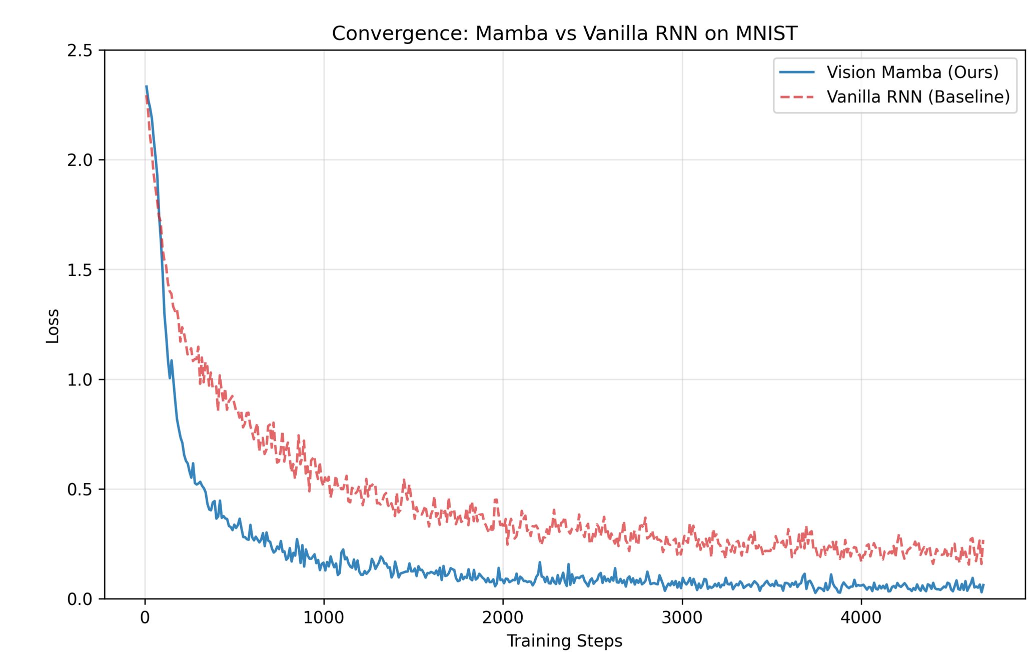

Figure 3: Training Loss Comparison on MNIST.

Model Architecture	Acc. (%)
Vanilla RNN	94.40
Causal Vision Mamba	97.00
Bi-Directional Vision Mamba (Ours)	**97.93**

Table 2: Classification Accuracy.

7. Results: Visulization of delta selection

We analyze the magnitude of the learned timestep Δ_t on a "Needle in a Haystack" task. The model exhibits distinct **activation spikes** (high Δ values) precisely at the positions of critical information (the "needle"), while maintaining low values for irrelevant noise. Mathematically, a large Δ_t increases the update rate of the state h_t, effectively **"opening the gate"** to write information into memory.

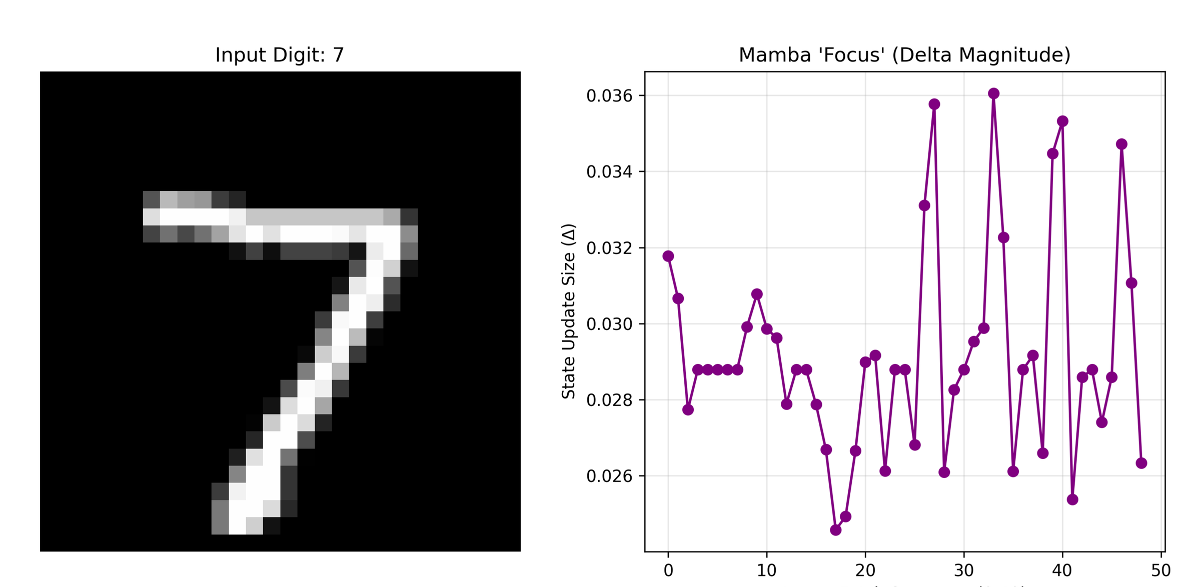

Figure 4: Visualization of Δ

8. Conclusion

- Successfully reimplemented **Mamba S6 from scratch** without CUDA kernels.
- **Bi-directional scanning** is crucial for non-causal data (images), boosting accuracy by $\sim$1% over causal Mamba.
- Demonstrated linear-time scaling potential on sequence tasks.

References

[1] Gu & Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*, 2023.