SONY

SF-Mamba: Rethinking State Space Model for Vision

Masakazu Yoshimura, Teruaki Hayashi, Yuki Hoshino, Wei-Yao Wang, and Takeshi Ohashi
(Sony Group Corporation)

Introduction

Mamba should be efficient than Attention

Mamba's theoretical computational cost is smaller

Attention FLOPs: $O(LD^2 + L^2D)$
Mamba FLOPs: $O(LD^2 + LD)$

L: token length
D: channel dimension

State Space Model (SSM)

$$H_t = A_t H_{t-1} + B_t x_t$$
$$Y_t = C_t H_t$$

Mamba: A_t, B_t, C_t as data-dependent

$$A_t = f_A(x_t), B_t = f_B(x_t), C_t = f_c(x_t)$$

But Vision Mamba models tend to be slow in processing low resolution images…

	MACs	224x224 Throughput [img/s]
Deit-S (Attention with Pytorch SDPA)	4.6G	4700
Vim-S (Mamba)	5.3G	1100

similar MACs but much slower

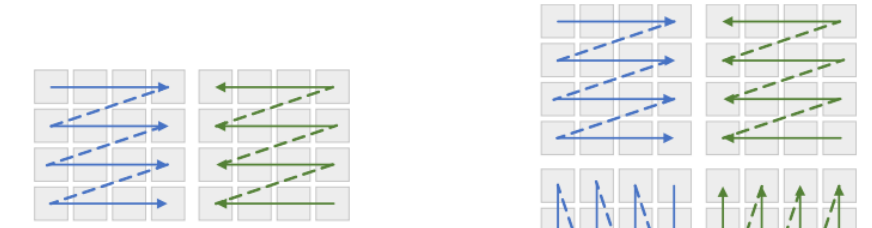
We achieve good accuracy-throughput trade-off

- by rethinking Vision Mamba from data flow perspective
  - data rearrangement is a problem → propose **auxiliary token swapping**
- by rethinking Vision Mamba from computational perspective
  - in-efficient data parallelism → propose **batch folding with periodic state reset**

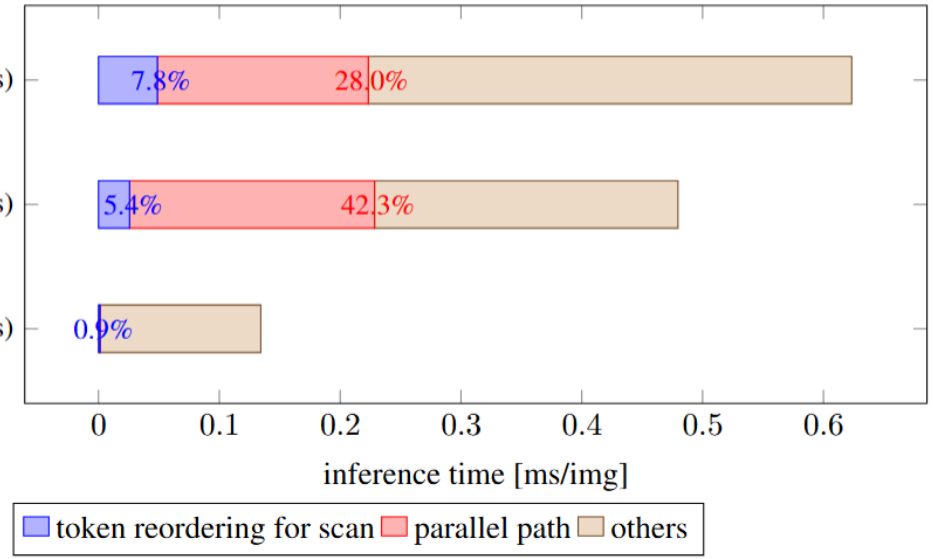
Rethinking Vision Mamba from Data Flow Perspective

Our Investigation

- Most previous Vision Mamba address causality constraints via multi-directional scans



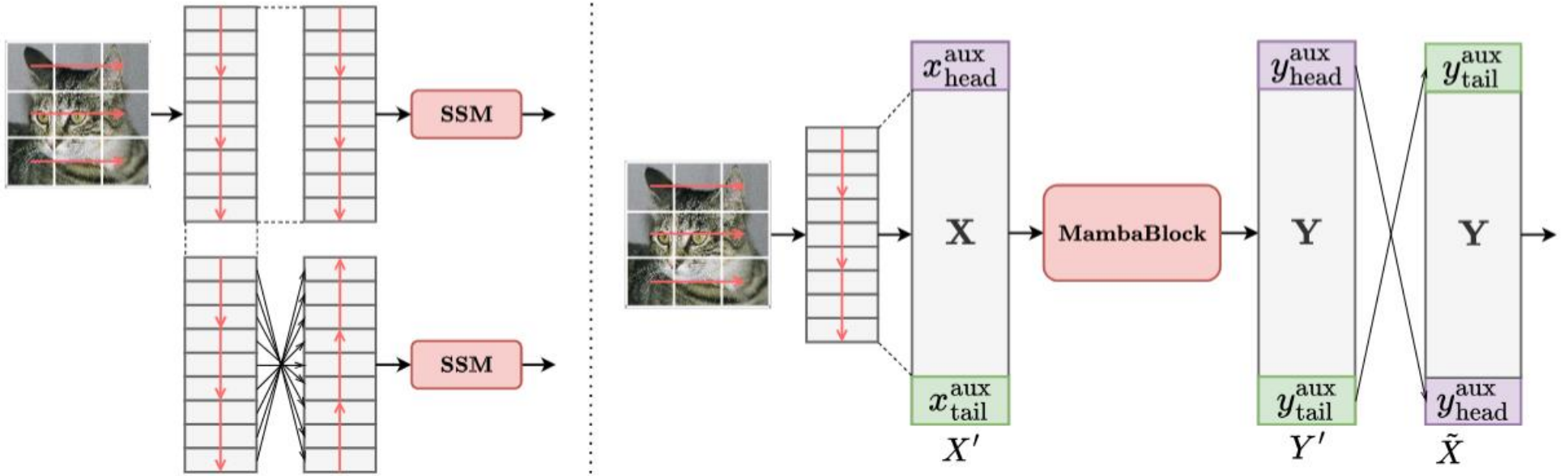
- However, token-reordering cost for multi-directional scan is unexpectedly large
- Moreover, the computational cost of the parallel paths naturally increases.



Rethinking Vision Mamba from Data Flow Perspective

Proposal

- Auxiliary token swapping with unidirectional scan
  - Two auxiliary tokens are introduced, and by swapping them between Mamba blocks, future-to-past information flow is achieved under unidirectional scan



Bi-directional Scan

T · 2 token length

O(n) permutation

Auxiliary Token Swap(ours)

T + 2 token length

O(1) permutation

Rethinking Vision Mamba from Computational Perspective

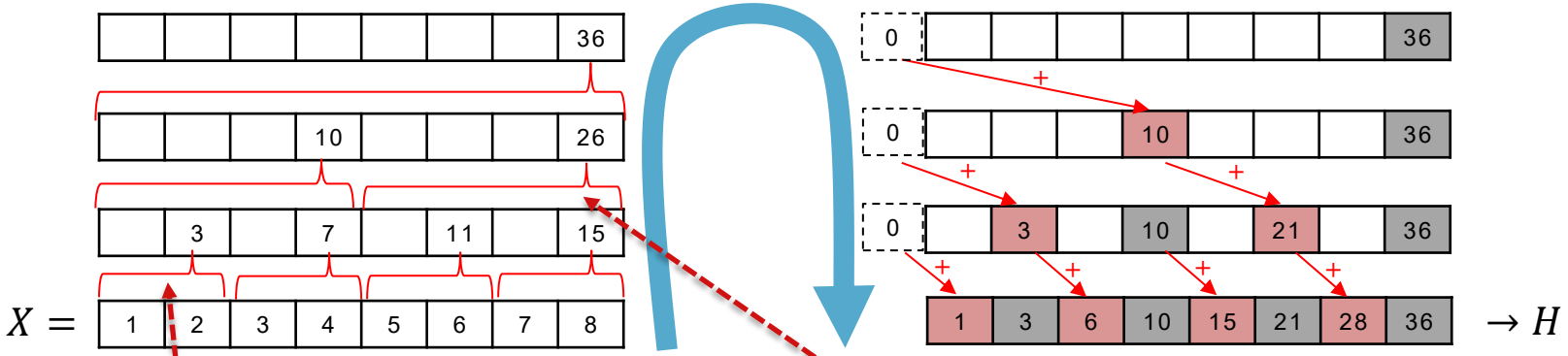
Our Investigation

- Mamba's **parallel scan** reformulates recurrent computation for parallel GPU execution
  - original recurrent computation: $H_t = A_t H_{t-1} + B_t X_t$

If $A = B = 1$ and $X =$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | → 1+2=3, 3+3= 6, 6+4=10 … (compute sequentially step by step)

$H =$ | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 |

  - parallel scan

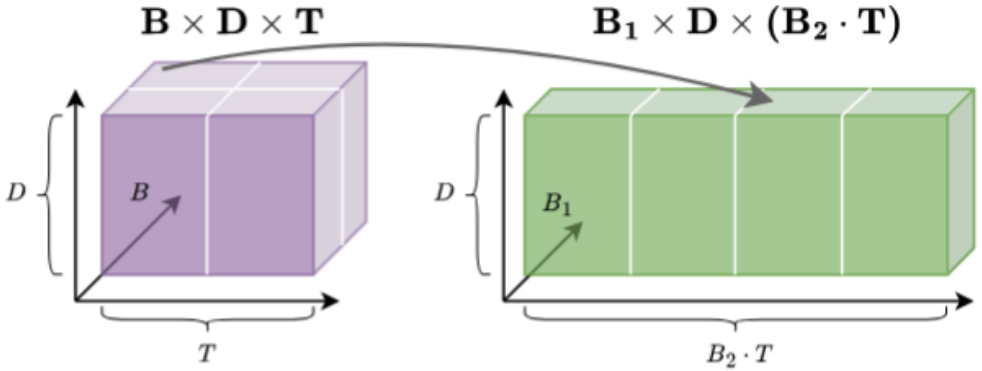


- However, GPU hardware constraint: Working with groups of 32 (or 64) threads
- Even with short token length, 32 GPU threads are launched but remain unused, leading to inefficiency

Rethinking Vision Mamba from Computational Perspective

Proposal

- Because Mamba's efficiency only holds for long token lengths, we move part of the batch dimension into the token dimension to artificially increase the effective sequence length



- Meanwhile, the state is periodically initialized to match the original computation

Periodic State Reset Trick

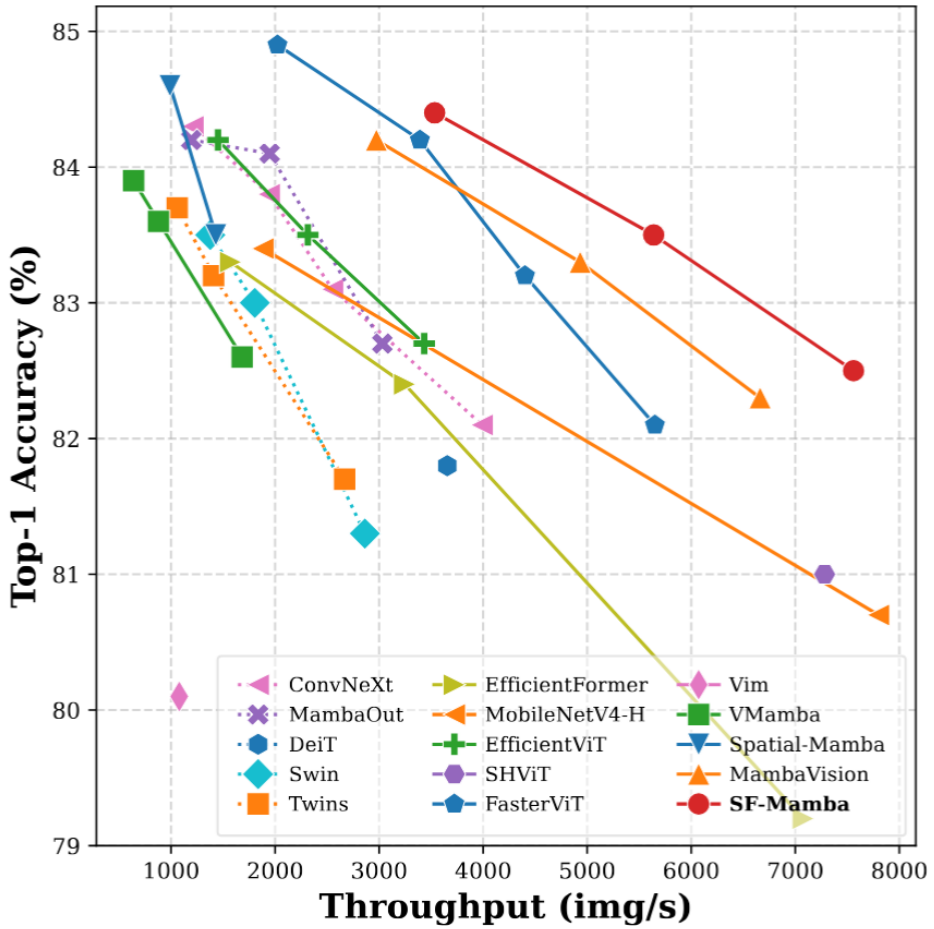
for $t = 1$ **to** $B_2 \cdot T$
  if $t \bmod T = 0$
    $A_t \leftarrow 0$
  $h_t \leftarrow A_t h_{t-1} + B_t x_t$
  $y_t \leftarrow C_t h_t$

$$H_t = A_t H_{t-1} + B_t x_t$$

$$Y_t = C_t H_t$$

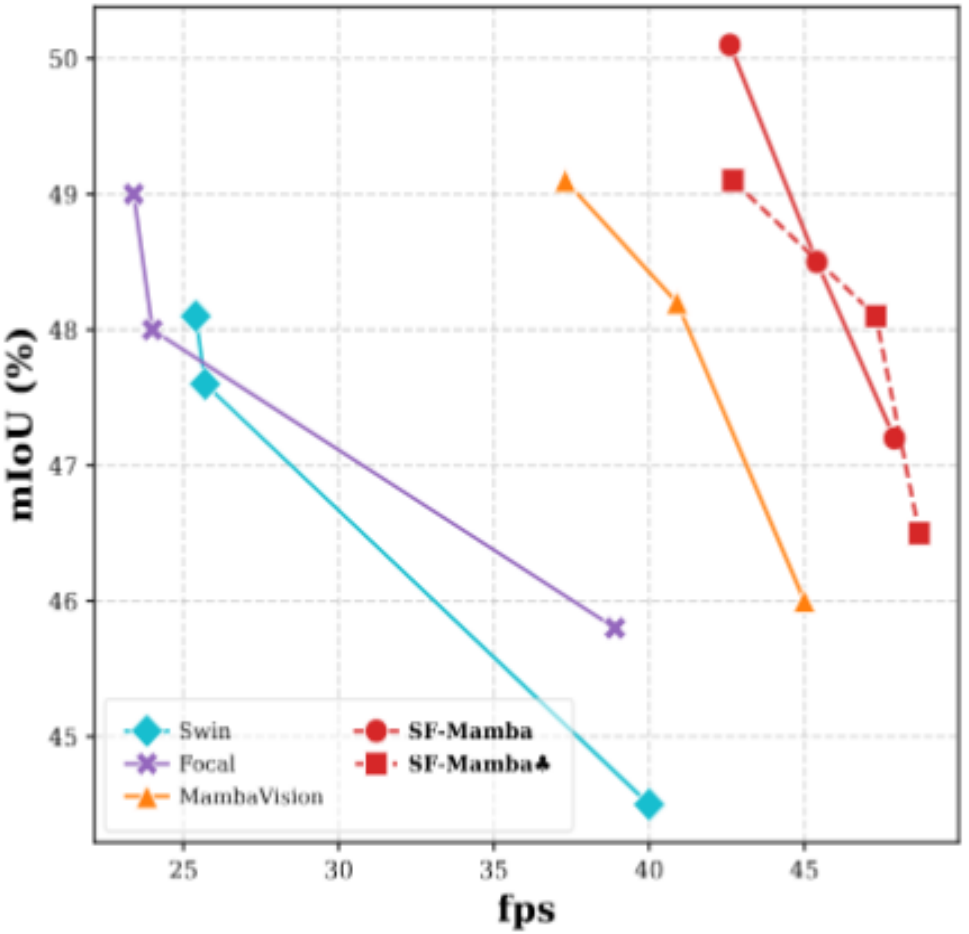
Evaluation

ImageNet-1K Image Classification



224x224 sized image , batch size 128

ADE20k Semantic Segmentation



1024x512 sized image, batch size 1

Experimental Setup

We implement our two proposals in Mamba block of MambaVision (CVPR25) architecture

Achieve better accuracy-throughput trade-off both on classification and segmentation

Conclusion

Conclusion

- We rethink recently successful Vision Mamba models to make them more efficient
- We found two problems and solve with two proposals
  - multi-directional scans reduce speed → **unidirectional scan with auxiliary token swapping**
  - Mamba's parallel scan is inefficient for short token length → **batch folding with periodic state reset**

Future Direction

- Enabling our method with Mamba-2, Mamba-3

The code is open-source!


SONY