SF-Mamba: Rethinking State Space Model for Vision

Masakazu Yoshimura [1] **Teruaki Hayashi** [2] **Yuki Hoshino** [1] **Wei-Yao Wang** [1] **Takeshi Ohashi** [1]

Abstract

Visual Mamba models have recently emerged as alternatives to Vision Transformers (ViTs), which suffer from quadratic complexity. While the recurrent scanning mechanism of Mamba offers computational efficiency, it inherently limits non-causal interactions between image patches. Prior works have attempted to address this limitation through various multi-scan strategies; however, these approaches suffer from inefficiencies due to suboptimal scan designs and frequent data rearrangement. Moreover, Mamba exhibits relatively slow computational speed under short token lengths, commonly used in visual tasks. In pursuit of a truly efficient vision encoder, we rethink the scan operation for vision and the computational efficiency of Mamba. To this end, we propose SF-Mamba, a novel visual Mamba with two key proposals: auxiliary patch **swapping** for encoding bidirectional information flow under an unidirectional scan and batch **folding** with periodic state reset for advanced GPU parallelism. Extensive experiments on image classification, object detection, and instance and semantic segmentation consistently demonstrate that our proposed SF-Mamba significantly outperforms state-of-the-art baselines while improving throughput across different model sizes. The source code is available at: `https://github.com/sony/SF-Mamba`.

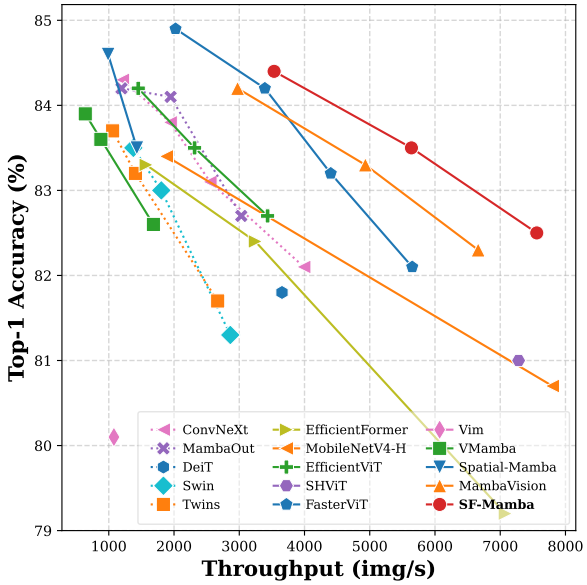


Figure 1. Top-1 accuracy and throughput on ImageNet-1K classification. SF-Mamba offers superior accuracy–throughput trade-offs compared to state-of-the-art architectures.

1. Introduction

The field of deep learning for vision has undergone several architectural shifts, driving progress across a wide range of applications including classification, segmentation, and detection (Deng et al., 2009; Krizhevsky et al., 2012; He et al., 2016; Simonyan & Zisserman, 2014; Ravi et al., 2025; Siméoni et al., 2025; Bolya et al., 2025). More recently, Vision Transformers (ViTs) (Dosovitskiy et al., 2020) have emerged as the dominant paradigm and offer strong flexibility and generalizability compared to convolution-based models by tokenizing images into patches and applying the self-attention mechanism (Vaswani et al., 2017). ViT-based vision models have been widely used for multi-modal learning tasks (Khan et al., 2022; Elharrouss et al., 2025); however, one of the main limitations of ViTs is the quadratic complexity for computing attention in terms of sequence length, which hinders the scalability to high-resolution inputs and large datasets with limited computational resources.

Mamba (Gu & Dao, 2023) introduces a selective state-space model (SSM), which enables data-dependent flexible token scanning in a left-to-right order, thereby achieving powerful but efficient processing with linear-time complexity. Building on its success, Mamba has been extended to the vision domain, achieving higher accuracy while being efficient in terms of memory cost, FLOPs, and the number of parameters (Zhu et al., 2024; Liu et al., 2024; Pei et al., 2025). In addition, recent studies suggest that visual Mamba has

[1]Sony Group Corporation, Tokyo, Japan [2]Sony Interactive Entertainment, Tokyo, Japan. Correspondence to: Masakazu Yoshimura <Masakazu.Yoshimura@sony.com>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

good transfer learning capabilities comparable to or even surpassing those of ViTs (Yoshimura et al., 2025; Galim et al., 2025), and it has the potential to replace the ViT-based foundation model ecosystem. However, many visual Mamba models suffer from slow processing speeds, especially on low-resolution images, which makes them not truly efficient. One reason is that Mamba adopts a recurrent left-to-right scanning mechanism, which prevents earlier patches from accessing information in future patches. As a result, many visual Mamba methods employ a multi-directional scan strategy, where the input sequence is rearranged and processed from multiple directions (e.g., from top-left to bottom-right, bottom-right to top-left). This allows models to compensate for the inability of standard Mamba to reference future patches and yields strong performance on vision tasks. However, such a rearrangement incurs substantial overhead during both training and inference. In fact, MambaVision (Hatamizadeh & Kautz, 2025) achieves a fast inference by avoiding costly multi-scan and instead rely on attention layers appended after the unidirectional scan. These attention layers enable information to flow backward, allowing earlier tokens to indirectly benefit from later tokens while retaining the efficiency of unidirectional Mamba. Yet, relying solely on attention for backward information flow poses a limitation: backward context can only be injected in deeper layers, leaving shallower layers deprived of future information and potentially restricting the expressiveness of the representation. Another reason why visual Mamba is slow lies in Mamba itself. As reported in the Mamba paper, unless the token length exceeds around 1000–2000, it is slower than Attention. Unless the task involves high-resolution images, the token length of the vision patches typically remains below 1000.

In this paper, we rethink visual Mamba from two perspectives in pursuit of a truly efficient image encoder. The first is the data flow. Instead of using the slower multi-directional scan, we adopt a unidirectional scan. However, this approach lacks future-to-past information flow, which is crucial to generate high-quality features. To address this, we propose an auxiliary patch swapping that enables future-to-past information flow within a unidirectional scan (Sec. 3.2). It introduces two additional tokens mixing the corresponding directional flow, which does not require significant burden compared with existing multi-scan approaches. The second perspective addresses the inefficiency of Mamba when processing short sequences. We attribute this limitation to suboptimal GPU parallelization, and to mitigate it, we introduce batch folding with periodic state reset (Sec. 3.3). This method reshapes batched inputs to maximize GPU thread utilization while preserving independence across sequences, thereby enhancing parallel efficiency. To this end, we propose **SF-Mamba**, which is equipped with the two key innovations of swapping and folding. Extensive experiments on image classification (Fig. 1), object detection, and semantic/instance segmentation demonstrate that SF-Mamba consistently outperforms state-of-the-art baselines while achieving faster inference, paving a new path toward efficient and effective Mamba architectures for vision.

In summary, our main contributions are three-fold:

- **Efficient uni-scan for non-causal ordering.** We propose a lightweight mechanism, auxiliary patch swapping, that introduces two learnable auxiliary tokens and a parameter-free swap operation, enabling bidirectional information flow across layers with negligible overhead compared to existing multi-scan approaches.
- **Efficient GPU parallelism for vision tasks.** To address inefficiency in low-resolution vision tasks, we design a batch folding strategy that merges the batch and sequence dimensions, maximizing GPU utilization while preserving the independence of hidden states across sequences. This method can speed up any Mamba-based method especially with short sequence processing.
- **Empirical validation across various tasks.** Experiments on classification, detection, and segmentation show that SF-Mamba outperforms state-of-the-art CNN-, Transformer-, hybrid CNN-Transformer-, and Mamba-based baselines.

2. Related Work

CNNs and Vision Transformers. Convolutional Neural Networks (CNNs) first led to breakthroughs in large-scale image classification (Deng et al., 2009; Krizhevsky et al., 2012), with deeper networks such as VGG (Simonyan & Zisserman, 2014) and ResNet (He et al., 2016) extending success to segmentation (Long et al., 2015) and detection (Ren et al., 2015). Vision Transformers (ViTs) (Dosovitskiy et al., 2020), inspired by self-attention (Vaswani et al., 2017), have since become the dominant paradigm by effectively modeling long-range dependencies. Follow-up works such as DeiT (Touvron et al., 2021), and Swin Transformer (Liu et al., 2021) improved efficiency and scalability. Models trained on large-scale data are used as a foundation for a variety of tasks (Oquab et al., 2024; Siméoni et al., 2025; Tschannen et al., 2025).

Visual State Space Models. To address the quadratic cost of attention, state-space models (SSMs) have emerged as efficient alternatives. Mamba (Gu & Dao, 2023) introduced selective state spaces, enabling linear-time complexity with strong long-range modeling. Inspired by this success, many visual Mamba variants (Zhu et al., 2024; Liu et al., 2024) extended SSMs to visual data.

Hybrid Architectures in Vision. Beyond single-paradigm designs, recent studies have demonstrated that hybrid

architecture can lead to more efficient encoding. The CNN-Transformer hybrids (Hatamizadeh et al., 2024; Li et al., 2022; Zheng, 2025) leveraged the local feature extraction and inductive biases of CNNs alongside the global context modeling of Transformers. More recently, Mamba–Transformer hybrids (Hatamizadeh & Kautz, 2025) have emerged, combining Mamba's computational efficiency with Transformers' receptive field. The hybrid architecture achieves superior efficiency–performance trade-offs and establishes state-of-the-art vision backbones.

Causality Constraint of Visual SSMs. From another perspective, visual SSMs face an inherent challenge: the *causality constraint*, which is also observed in vision-language models (Wang et al., 2025c). Since state-space models process inputs sequentially, each hidden state only depends on the past, preventing access to the global spatial context. Many visual Mamba methods address causality constraints via multi-directional scans. Some approaches like Vim (Zhu et al., 2024) and Mamba-R (Wang et al., 2025a) adopt bi-directional scans, while recent models are based on cross-scan (Liu et al., 2024), which performs bi-directional scan along both horizontal and vertical axes, totaling four directions to better capture image structure. Variants such as GroupMamba (Shaker et al., 2025), MSVMamba (Shi et al., 2024), EfficientVMamba (Pei et al., 2025), and DefMamba (Liu et al., 2025) enhance cross-scan through zigzag patterns, multi-resolution scan, atrous sampling, or deformable directions. Despite being parameter-efficient, multi-directional scans are slow. Cross-scan based methods are particularly suffer from slow speed due to increased FLOPs from four parallel scans and costly data rearrangement between 2D formats (for 2D convolution) and 1D formats (for scanning). Rearranging tokens for four directions adds further overhead, especially in vertical scans, which involve scattered memory access. While bi-directional scan avoids 2D/1D format switching, it still requires rearranging data for the backward scan and maintaining two parallel paths. The preliminary experiments in Appendix (Fig. 6) show that the multi-scan strategy actually causes a significant degradation in inference speed.

A recent study, Adventurer (Wang et al., 2025b), tackles the causality constraint of Mamba2 (Dao & Gu, 2024) using series bi-directional scans, which alternate scan directions between layers. It also inserts a globally averaged token in every layer to facilitate limited context exchange of series bi-directional scan. While this mechanism only requires a single scan in each block, it requires explicit flipping operations with $O(n)$ permutation cost and an additional averaging cost, resulting in reduced throughput.

Several methods are starting to achieve high accuracy with unidirectional scan. Spatial-Mamba (Xiao et al., 2025) uses 2D atrous convolution with a wide receptive field to access future patches, although the 2D/1D format switching degrades the speed. MambaVision (Hatamizadeh & Kautz, 2025) incorporates Attention in later layers to capture global context. However, relying solely on attention for future-to-past information flow might not be optimal.

Furthermore, although previous methods adopt Mamba due to its parameter efficiency and superior accuracy, it remains slower than Attention for token lengths below 1000 to 2000 (Gu & Dao, 2023). These limitations motivate the development of a truly efficient visual Mamba.

3. Method

3.1. Preliminaries

Mamba State Space Model. Mamba (Gu & Dao, 2023) is a selective state space model (SSM) that processes a sequence $X = (x_1, \ldots, x_T)$ by recurrently updating a hidden state h_t:

$$h_t = A_t h_{t-1} + B_t x_t, \qquad y_t = C_t h_t, \tag{1}$$

where h_t is the hidden state, y_t the output, and A_t, B_t, C_t are input-dependent matrices.

In the vision setting, an image is divided into T patches, each embedded as $x_t \in \mathbb{R}^D$, forming a sequence $X = (x_1, \ldots, x_T)$. A batch of such sequences is denoted as $X_{\text{in}} \in \mathbb{R}^{B \times T \times D}$, where B is the batch size, T the number of patches (sequence length), and D the embedding dimension. In this case, Mamba can be viewed as a mapping $f_\theta : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ that applies the recurrence in Equation (1) to patch sequences.

Using a *parallel scan* algorithm, this recursive operation is efficiently computed. Specifically, Mamba uses a warp scan function (NVIDIA, 2025a) implemented in the CUDA backend, which enables high-speed parallel scan by allowing multiple threads to share data through the fast SRAM memory of the GPU. Since this warp scan function operates in groups of 32 threads, each sequence must be processed using at least 32 threads. Note that this is not a constraint of the operation itself, but a constraint imposed by modern GPU hardware. Any operations must be executed in units of 32 threads internally.

Mamba-Transformer Hybrid Architecture. We employ a Mamba-Transformer hybrid architecture, because previous studies indicate that the hybrid architecture achieves promising efficiency. In other words, we employ MambaVision (Hatamizadeh & Kautz, 2025) architecture as a macro level. It uses a four-stage hierarchical design. The first two stages are CNN-based and serve as a kind of deep patch embedding. The latter two stages consist of several Mamba blocks followed by several Attention Blocks. MambaVision accelerates processing by adopting a simple unidirectional

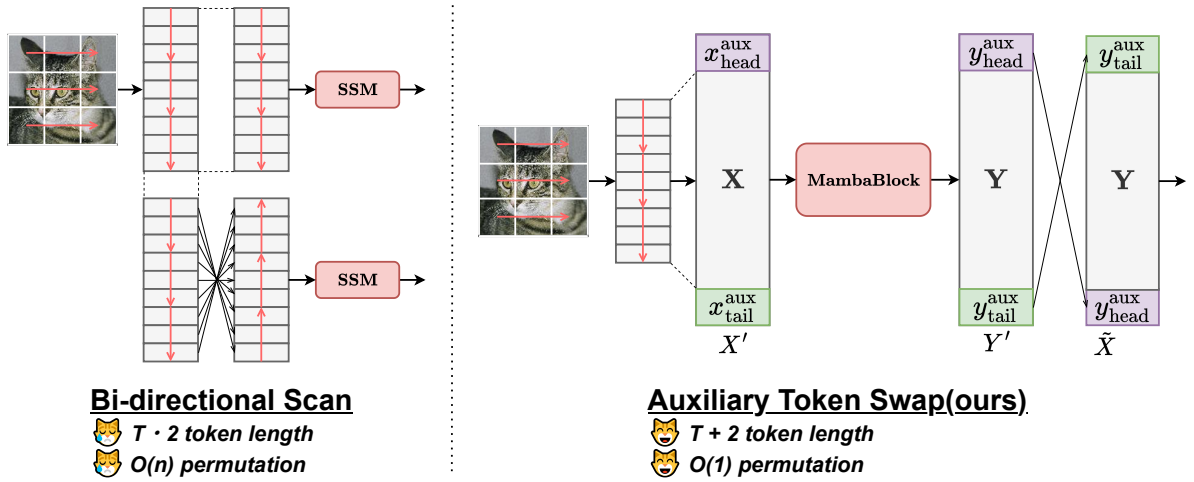


Figure 2. **Future-to-Past Information Routing via Auxiliary Token Swapping.** The left figure illustrates why the commonly used multi-directional scan in visual Mamba fails to achieve high speed, while the right figure presents our proposed solution. We prepend/append learnable auxiliary tokens to the patch sequence $x^{\text{aux}}_{\text{head}}$ and $x^{\text{aux}}_{\text{tail}}$. Within each MambaVision block, the causal selective scan aggregates sequence-wide context into the tail token $y^{\text{aux}}_{\text{tail}}$. A lightweight, parameter-free Swap operation then moves this global summary to the sequence head, yielding $\tilde{X}$ for the next layer such that all patch states are conditioned on global context. It incurs negligible computational overhead while enabling effective global-context propagation across layers.

scan. However, due to the *causality constraint*, it cannot reference future patches from past ones, so future-to-past information flow relies on subsequent Attention blocks. Detailed structure and formulation of the MambaVision-based blocks are provided in Appendix B.1.

3.2. Rethinking Visual SSM from Data Flow Perspective

Future-to-Past Information Routing via Auxiliary Token Swapping. Since image patches do not exhibit a strict causal ordering, restricting Mamba blocks to a unidirectional scan can be limiting: tokens in earlier regions (e.g., the top-left) cannot directly access information from later regions (e.g., the bottom-right), which hinders representation learning. While multi-scan approaches such as bidirectional or cross-scan alleviate this issue, they require repeated reordering of the data, which introduces substantial computational overhead and complicates implementation. Hence, we propose a future-to-past information flow with minimal additional cost by introducing two *auxiliary tokens*.

At the first Mamba block in each stage, the two auxiliary tokens, $x^{\text{aux},1}_{\text{head}}$ and $x^{\text{aux},1}_{\text{tail}}$, are initialized as data-dependent values (i.e., $x^{\text{aux},1}_{\text{head}} = x^{\text{aux},1}_{\text{tail}} = avg(X)$, where $avg(\,)$ averages the sequential dimension). The tokens are then concatenated at both ends of the input X for the first Mamba block:

$$X' = (x^{\text{aux},1}_{\text{head}},\ x_1, \ldots, x_T,\ x^{\text{aux},1}_{\text{tail}}). \tag{2}$$

After processed by the i-th Mamba block, we swap the two tokens for the input of the next Mamba block (see Fig. 2):

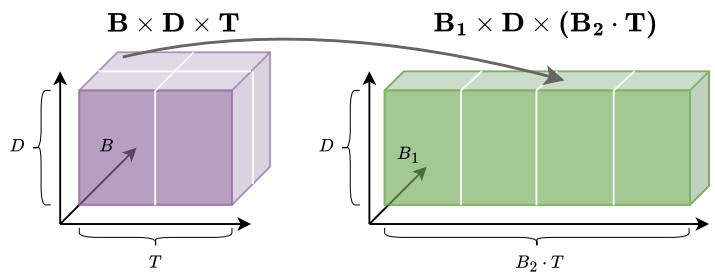
$$x^{\text{aux},i+1}_{\text{head}} = y^{\text{aux},i}_{\text{tail}},\ \ x^{\text{aux},i+1}_{\text{tail}} = y^{\text{aux},i}_{\text{head}}, \tag{3}$$

where $y^{\text{aux},i}_{\text{head}}$ and $y^{\text{aux},i}_{\text{tail}}$ are the output tokens with respect to $x^{\text{aux},i}_{\text{head}}$ and $x^{\text{aux},i}_{\text{tail}}$. By training this architecture, we expect that $y^{\text{aux},i}_{\text{tail}}$ extracts the necessary information from all tokens in the i-th layer, and $y^{\text{aux},i}_{\text{head}}$ serves as a feature that determines how $y^{\text{aux},i+1}_{\text{tail}}$ should be extracted in the next layer. Then, by swapping as shown in Eq. 3, the patch tokens of the next layer (x_1, x_2, ..., X_T) can refer to $x^{\text{aux},i+1}_{\text{head}}$, which contains features from all positions, allowing future-to-past information routing. This intended operation is natural for the selective scan SSM and does not disrupt the original mechanism, which selectively extracts the necessary information as y_t from hidden states that span from $t = 0$ to $t = t$. Similarly, we expect it selectively extracts the necessary information as $y^{\text{aux},i}_{\text{tail}}$ from hidden states that span from $t = 0$ to $t = T$.

In contrast to multi-scan strategies, our approach does not rely on multiple parallel paths or global token rearrangements. Instead, it swaps only two tokens within the sequence, introducing negligible computational overhead.

3.3. Rethinking Visual SSM from Computational Perspective

Batch Folding with Periodic State Reset. We identify that Mamba's inefficiency in low-resolution vision tasks arises from the warp-scan implementation, which achieves high throughput by utilizing 32 GPU threads per sequence (Sec. 3.1). In vision models, however, the number of patches (i.e., sequence length) is relatively small (e.g., 196 and 49 for MambaVision Stage 3 and 4), making the allocation of 32 threads per sequence highly underutilized and inefficient.



Periodic State Reset Trick

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

Figure 3. **Batch folding with periodic state reset.** (**Left**) An input tensor of shape $[B, D, T]$ is reshaped into $[B_1, D, (B_2 \cdot T)]$, concatenating B_2 short sequences into a longer one. This reshaping mixes hidden states across batches. (**Right**) To avoid information leakage, we reset the recurrence every T steps. Since $h_t \leftarrow A_t h_{t-1} + B_t x_t$, setting $A_t = 0$ at boundaries is equivalent to re-initializing the hidden state. In contrast, B_t (input projection) and C_t (output projection) operate locally and therefore remain unchanged.

To address this, we propose a batch folding strategy that reshapes the input by merging the batch dimension into the sequence dimension (Fig. 3, left). This improves parallel efficiency in scenarios with many short sequences while preserving the correctness of the computation. Let $Z \in \mathbb{R}^{B \times D \times T}$ denote the batched tokens before entering the SSM. We reshape Z into

$$Z' \in \mathbb{R}^{B_1 \times D \times (B_2 \cdot T)}, \quad B = B_1 \cdot B_2, \tag{4}$$

which concatenates B_2 short sequences into one longer sequence. This operation is a bijective permutation of indices, so the original tensor can be exactly recovered. Intuitively, this extends the effective sequence length in a pseudo manner, allowing the parallel scan to operate more efficiently by reducing kernel launch overhead and reducing inefficient use of memory bandwidth.

However, this reshaping mixes hidden states across different sequences. To preserve independence, we effectively use and improve a computational trick implemented in vLLM software (Kwon et al., 2023), which was originally devised for Mamba inference in LLMs to handle multiple sequences of varying lengths without padding. Our trick for preserving the dependence of the folded data named *periodic state reset trick* is as follows (Fig. 3, right). In every T step, we set $A_t = 0$, which removes dependence on h_{t-1} and resets the hidden state. Then, all the hidden states become identical to those without batch folding. By unfolding the output, the output becomes equivalent to that obtained without applying batch folding. Note that B_t and C_t act only on the current input and hidden state, respectively, and thus do not require resetting. Since it only resets A, there is only a minimal increase in processing time.

Adaptive B_1. In batch folding, it is not optimal to increase the virtual sequence size indefinitely. The ideal ratio between B_1 and B_2 is determined in a complex manner based on factors such as batch size B, number of input tokens T, model dimension D, state dimension S, and the number of threads used when invoking CUDA. Therefore, we precompute and store combinations of (B, D, T, S), along with the optimal B_1/B ratio, in a coarse-grained 4-dimensional lookup table (LUT). At runtime, we retrieve the optimal B_1 value from this LUT as follows:

$$B_1 = f(B, B \cdot \mathrm{LUT}(B, D, T, S)), \tag{5}$$

where $f(a, b)$ is a function that returns a divisor of a, which is closest to b.

1-D Depthwise Convolution for Batch Folded Data. Although batch folding improves the speed of the SSM component, the reshaping operation in Eq. 4 introduces a slowdown. To mitigate this, we apply the transformation in Eq. 4 only at the initial Mamba block of each stage, and then continue computation using the batch-folded tensor shape. Since the Linear and LayerNorm layers operate per token, they do not pose any issues. However, the 1D depthwise convolution in the Mamba block presents a challenge. To address this, we implement a convolution that supports batch-folded data, ensuring that no convolution occurs across the boundaries between T sequences. In other words, our convolution CUDA kernel performs implicit padding at the boundary of each T sequence.

4. Experiments

We conduct comprehensive experiments to evaluate SF-Mamba across three fundamental computer vision tasks: image classification, semantic segmentation, object detection with instance segmentation (Appendix C.4 and C.5). Our experimental setup follows the protocols established by previous works (Liu et al., 2024; Xiao et al., 2025; Hatamizadeh & Kautz, 2025) to ensure fair comparisons. We evaluate three model variants (T/S/B) with different scales to analyze the accuracy-throughput trade-offs. For all downstream tasks, we use models pre-trained on ImageNet-1K as backbones. Detailed training configurations and hyperparameters are provided in Appendix A.

Table 1. Detailed comparison of image classification performance on ImageNet-1K. All models are evaluated with 224×224 input resolution. In the token mixer type, C, P, A, and S denote Convolution, Pooling, Attention, and SSM, respectively.

Model	Mixer	Params	MACs	img/s	Acc. (%)
ConvNeXt-T	C	29M	4.5G	3990	82.1
MambaOut-T	C	27M	4.5G	3031	82.7
Swin-T	A	29M	4.5G	2863	81.3
Twins-S	A	24M	2.9G	2669	81.7
EfficientFormer-L3	P+A	31M	3.9G	3246	82.4
FasterViT-0	C+A	31M	3.3G	5651	82.1
Vim-S	S	26M	5.3G	1079	80.1
VMamba-T	C+S	30M	4.9G	1684	82.6
Spatial-Mamba-T	C+S	27M	4.5G	1430	83.5
MambaVision-T	C+S+A	32M	4.4G	6662	82.3
SF-Mamba-T	C+S+A	32M	4.5G	**7600**	82.5
ConvNeXt-S	C	50M	8.7G	2552	83.1
MambaOut-S	C	49M	9.0G	1948	84.1
Swin-S	A	50M	8.7G	1805	83.0
Twins-B	A	56M	8.6G	1409	83.2
FasterViT-1	C+A	53M	5.3G	4402	83.2
EfficientViT-B3	C+A	49M	4.0G	2315	83.5
VMamba-S	C+S	50M	8.7G	879	83.6
Spatial-Mamba-S	C+S	43M	7.1G	990	84.6
MambaVision-S	C+S+A	50M	7.5G	4933	83.3
SF-Mamba-S	C+S+A	50M	7.6G	**5639**	83.5
ConvNeXt-B	C	89M	15.4G	1943	83.8
MambaOut-B	C	85M	15.9G	1195	84.2
Swin-B	A	88M	15.4G	1377	83.5
Twins-L	A	99M	15.1G	1059	83.7
EfficientFormer-L7	P+A	82M	10.2G	1573	83.3
FasterViT-2	C+A	76M	8.7G	3392	84.2
VMamba-B	C+S	89M	15.4G	640	83.9
Spatial-Mamba-B	C+S	96M	15.8G	670	85.3
MambaVision-B	C+S+A	98M	15.0G	2974	84.2
SF-Mamba-B	C+S+A	98M	15.1G	**3534**	84.4

4.1. Image Classification

Experimental Setup. We first evaluate our models on image classification task using ImageNet-1K (Deng et al., 2009), which contains 1.28M training images and 50K validation images across 1,000 categories. Models are trained from scratch for 300 epochs following prior works (Hatamizadeh & Kautz, 2025; Liu et al., 2024). Throughput is measured on a single NVIDIA A100 GPU with a batch size of 128 (see Appendix A.4 for details).

Results. As shown in Fig. 1, SF-Mamba achieves superior efficiency-accuracy trade-offs with consistent improvements across all model scales (T/S/B variants) compared to existing architectures including CNN-based models (ConvNeXt (Liu et al., 2022), MambaOut (Yu & Wang, 2025)), Transformer-based models (DeiT (Touvron et al., 2021), Swin (Liu et al., 2021), Twins (Chu et al., 2021)), hybrid CNN-Transformer architectures (EfficientFormer (Li et al., 2022), EfficientVit (Cai et al., 2023), MobileNetV4-H-M (Qin et al., 2024), SHViT (Yun & Ro, 2024), FasterViT (Hatamizadeh et al., 2024)), and recent Mamba-based models (Vim (Zhu et al., 2024), VMamba (Liu et al., 2024), Spatial-Mamba (Xiao et al., 2025), MambaVision (Hatamizadeh & Kautz, 2025)). Note that some models in Fig. 1 have non-224×224 input resolution (see Appendix A for details). Table 1 provides a detailed comparison of models evaluated at the standard 224×224 resolution.

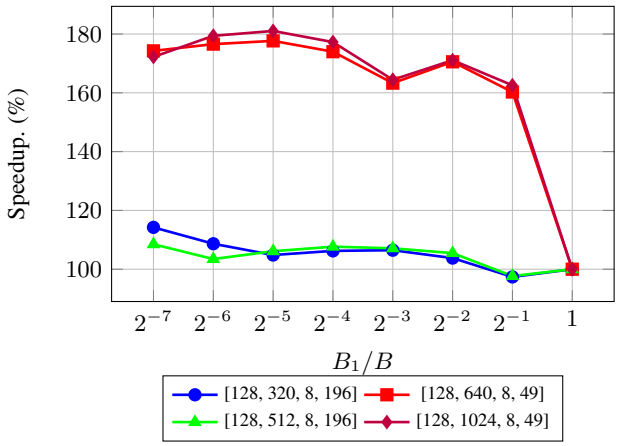


Figure 4. How much we can speedup the SSM calculation by changing B_1. The four configurations of [batch size, dimension, state dimension, sequence length] are exact settings for ours-T stage 3, ours-T stage 4, ours-B stage 3, ours-B stage 4.

Analysis. To analyze the effect of *batch folding with periodic state reset*, the speed of the SSM kernel part is measured, as shown in Fig. 4. A clear speedup of 110% to 180% is observed when the batch dimension is virtually shifted into the sequential dimension. This improvement is especially significant when the sequential length is short. The reason lies in the CUDA parallel scan algorithm, which plays a crucial role in Mamba's speedup. This algorithm requires at least 32 threads per sequence, and when the sequence is short, the overhead of allocating 32 threads becomes substantial. By virtually extending the sequence length, we can utilize the allocated threads more efficiently, leading to a significant performance boost.

Table 2. The computational speedup achieved by our method

impl. opt.	BFold	B_1	conv	img/s
				6662
✓				6989
✓	✓	1	✓	7601
✓	✓	4	✓	7641
✓	✓	adaptive		7279
✓	✓	adaptive	✓	**7685**

Next, we evaluate how much our proposed method can improve the overall model speed, as shown in Table 2. Our baseline is MambaVision-T, and we measure the degree of speed improvement from it. First, we improve the implementation and observe a speedup, which we denote as

impl. opt. This improvement primarily stems from the SSM CUDA kernel, which we rewrote based on the Mamba SSM CUDA kernel to suit our method. Building upon this, our *batch folding with periodic state reset (BFold)* technique achieves a significant speedup. Furthermore, our *adaptive* B_1 approach, which adaptively adjusts B_1 according to input and weight sizes, enables further improvements in inference speed. Using our 1-D convolution compatible with batch-folded data makes it unnecessary to repeatedly convert the data back to the standard format, resulting in improved speed. Since MambaVision is a hybrid model that combines Attention and Mamba, it goes without saying that it does not achieve the same level of speed improvement as described in Fig. 4. However, it still delivers significant performance gains compared to the baseline.

Table 3. The effectiveness of auxiliary token swapping and its ablation results.

swap	aux. token	discard timing	IN1K	ADE20K	img/s
			82.2	46.0	7645
	learnable	before attn	82.1	46.2	7613
✓	learnable	before attn	82.3	46.5	7585
✓	data-dependent	before attn	82.4	46.8	7602
✓	data-dependent	after 1st attn	**82.5**	**47.2**	7600
✓	data-dependent	after attn	82.4	46.6	7597

Table 3 presents an ablation study on *auxiliary token swapping*. The swapping improves performance with only a minimal impact on inference speed. Simply adding learnable tokens without performing swapping degrades performance, indicating that the improvement does not come from the increased flexibility provided by the additional tokens, but rather from the bidirectional information flow enabled by swapping. Looking at Fig. 7 in the Appendix, we can clearly see that it indeed achieves substantial bi-directional information propagation. Initializing auxiliary tokens as globally averaged data-dependent values proves more effective than employing a learnable token, which is commonly used as a class token(Dosovitskiy et al., 2020; Zhu et al., 2024). In addition, initializing this token with global features may allow subsequent layers to effectively acquire the global information needed for the next layer. As for where to discard this token, the most efficient approach is to remove it after the first attention layer.

Table 4 demonstrates which scan method is most efficient. Here, we evaluate with two macro-architectures. One is MambaVision-T, and the other is an architecture in which all attention blocks in MambaVision-T are replaced into Mamba blocks. The parallel bidirectional scan (bi-scan) requires twice the SSM computation cost due to its parallel nature, resulting in inefficient computation. Even in parallel bi-scan (cat), which halves the channel dimension to align MACs, the speed is slow due to the tensor rearrangement cost.

In contrast, the series bidirectional scan flips the token sequence at each layer, with odd-numbered blocks scanning in the forward direction and even-numbered blocks scanning in the reverse direction. This design allows for the creation of global features without increasing the number of FLOPs. However, the accuracy improvement is not as significant as expected. We hypothesize that DropOut (Huang et al., 2016), which is effective in preventing overfitting and gradient vanishing, may not be compatible with the series bidirectional scan architecture, which has an asymmetric structure across layers. To this end, Adventurer (Wang et al., 2025b) style model does not achieve high accuracy, although the introduced global averaged token actually improves from the normal series bi-scan setting. Also, the flipping operation needed for the bi-scan the block incurs the speed with an $O(n)$ permutation cost.

On the other hand, our *auxiliary token swapping* only swaps two tokens, minimizing the slowdown while achieving comparable or superior accuracy. The fact that the swapping improves a lot from unidirectional scan with the Mamba only architecture indicates that it allows future-to-past token information flow with the swapping, thereby facilitating the creation of better features (See Fig. 7 in Appendix).

We also present the robustness of our proposed methods by applying them to Vim (Zhu et al., 2024) macro-architecture in Table 13 in Appendix.

4.2. Semantic Segmentation

Experimental Setup. We evaluate on ADE20K (Zhou et al., 2017) using UperNet (Xiao et al., 2018) as the segmentation framework. This task requires assigning a semantic class label to each pixel in the image across 150 categories. Models are trained with 512×512 crop resolution following standard protocols (Liu et al., 2024; Xiao et al., 2025). Performance is measured by mean Intersection over Union (mIoU) (Csurka et al., 2013).

Results. Fig. 5 summarizes the semantic segmentation performance on ADE20K. Segmentation, unlike image classification, requires both fine-grained pixel-level boundary detection and global structural understanding to accurately identify object classes. Therefore, enabling the Mamba block to incorporate future patch information through state swapping significantly improves the accuracy compared to the baseline MambaVision. During inference, the model processes images at a resolution of 512×2048, which differs from the training resolution. To accommodate this discrepancy, both Mamba and Attention are implemented to process per windowed region, where the window size matches the training image dimensions (see Appendix B.3 for details). Therefore, although the inference speed is mea-

Table 4. Comparison of effective scan designs. The "series bi-scan + gap token" follows Adventurer (Wang et al., 2025b), where a global token obtained via global average pooling is used. In parallel bi-scan, "cat" splits the input along channels, applies bi-scan, and concatenates the results, while "add" duplicates the input, applies bi-scan, and sums the outputs. The "Vim block" indicates exact Vim block is used instead of making MambaVision block bi-directional.

macro-arch.	MambaVision-T				MambaVision-T w/o Attention			
scan	Params	MACs	img/s	acc.	Params	MACs	img/s	acc.
uni-scan	31.8M	4.4G	6979	82.2	29.4M	4.2G	6238	80.2
series bi-scan	31.8M	4.4G	6911	82.3	29.4M	4.2G	6113	80.4
series bi-scan+gap token (Adventurer)	31.8M	4.5G	6856	82.3	29.4M	4.3G	6027	80.7
parallel bi-scan (cat)	31.8M	4.4G	6834	82.2	29.4M	4.2G	5987	80.8
parallel bi-scan (add)	31.9M	4.5G	6235	82.3	29.7M	4.3G	5138	81.1
parallel bi-scan (add) (Vim block)	33.5M	4.6G	4612	82.4	32.8M	4.6G	3256	**81.7**
uni-scan + swap (ours)	31.8M	4.5G	**6926**	**82.5**	29.4M	4.3G	**6171**	81.0
uni-scan + swap (ours) + Bfold	31.8M	4.5G	7600	82.5	29.4M	4.3G	7306	81.0

sured with a batch size of 1, the model still benefits from batch folding based on the number of windows, resulting in an improved speed. Our base size model is faster compared to the Tiny versions of Swin and Focal Transformer, while achieving over 4 points higher mIoU. The best trade-off of SF-Mamba among recent visual backbones indicates that the proposed auxiliary patch swapping improves both efficiency and generalization capability, offering superior accuracy-cost trade-off.

The SF-Mamba♣ configuration in Table 6 adopts more granular window attention to reduce the quadratic computational cost of Attention by dividing the image into smaller windows. Meanwhile, we do not use a window size smaller than the training image size for the Mamba blocks to capture global context, resulting in better efficiency in terms of FLOPs as shown in Table6. Because windowing incurs a reshape overhead, the inference speed of SF-Mamba and SF-Mamba♣ is comparable on an A100 GPU. However, in environments where FlashAttention2 cannot be used and the quadratic cost of Attention is directly incurred, SF-Mamba♣ becomes particularly important. The nature of Mamba being not affected by long tokens is the explicit merit of Mamba over Attention. More details can be found in Appendix C.6. Furthermore, Table 14 in Appendix indicates that much larger input images benefit a large speed gain in SF-Mamba♣ configuration.

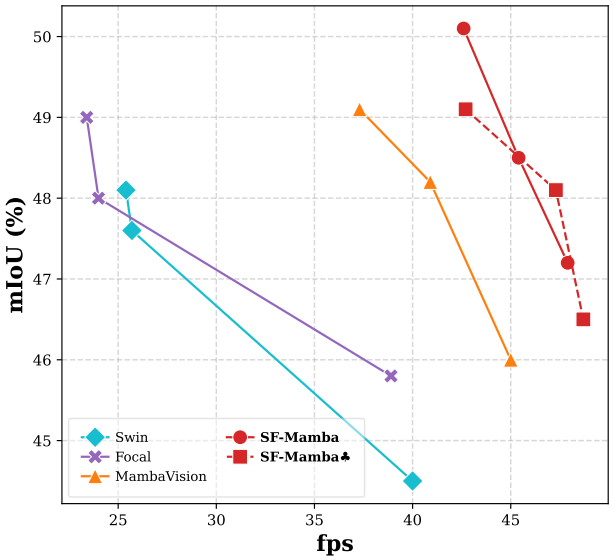


Figure 5. **Throughput–accuracy trade-off on ADE20K.** The x-axis denotes frames per second with batch size 1 setting (higher is better), and the y-axis shows mIoU (higher is better). SF-Mamba variants lie on the Pareto front.

5. Conclusion

In this paper, we rethink the recently effective visual Mamba approach from two perspectives. The first is an efficient scanning method for vision tasks. Previous studies have addressed the causality constraint of SSM by introducing multiple scan directions, but this comes with a significant drop in inference speed. To overcome this, we propose auxiliary token swapping, which enables future-to-past information flow without sacrificing inference speed, thereby achieving efficient scanning. The second perspective investigates why Mamba tends to be slow in image processing. We identified the bottleneck and proposed batch folding, a method that virtually extends the sequence length while keeping the identical SSM output, resulting in faster processing without accuracy drop. SF-Mamba, a novel Mamba-based framework with these proposals, achieves a superior accuracy-speed trade-off compared to existing methods. Although the latter technique may not provide benefits during inference with batch size = 1, training typically uses batch size > 1, so the speed-up advantage is expected in most training scenarios. Moreover, even with a batch size of 1, Mamba-based approaches–such as those employing local windows as in our segmentation experiments or multi-directional scan–result in an effective batch size larger than 1 for the SSM, thereby allowing for performance acceleration. We believe that this work will advance the development of efficient and effective image recognition models.

Impact Statement

Our work aims to improve the computational efficiency of state-space models for vision tasks, which has potential benefits for both large-scale and resource-constrained deployment scenarios. The proposed swapping and batch-folding mechanisms offer improved throughput at low resolution and ultra-high resolution (See Table 14). This may reduce training and inference costs in high-resolution applications such as medical imaging, aerial monitoring, and robotics.

Although we were unable to include evaluations on edge GPUs or mobile hardware in this submission, prior studies (Wei, 2025; NVIDIA, 2025b) have shown that Mamba kernels can be deployed on devices such as NVIDIA Jetson and iOS through optimized runtimes (e.g., TensorRT, mobile accelerators). Our method should be adaptable to these platforms since the same selective scan is used in our core algorithm. Enabling efficient state-space model inference on edge devices may broaden access to low-power real-time vision systems, but also calls for careful consideration of responsible deployment practices in safety-critical or privacy-sensitive contexts.

References

Bolya, D., Huang, P., Sun, P., Cho, J. H., Madotto, A., Wei, C., Ma, T., Zhi, J., Rajasegaran, J., Rasheed, H., Wang, J., Monteiro, M., Xu, H., Dong, S., Ravi, N., Li, D., Dollár, P., and Feichtenhofer, C. Perception encoder: The best visual embeddings are not at the output of the network. *CoRR*, abs/2504.13181, 2025.

Cai, H., Li, J., Hu, M., Gan, C., and Han, S. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17256–17267, 2023.

Cai, Z. and Vasconcelos, N. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43 (5):1483–1498, 2019.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., and Shen, C. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9355–9366, 2021.

Contributors, M. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

Csurka, G., Larlus, D., Perronnin, F., and Meylan, F. What is a good evaluation measure for semantic segmentation? *Proceedings of the British Machine Vision Conference*, pp. 32.1–32.11, 2013.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Dao, T. and Gu, A. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ding, X., Zhang, X., Han, J., and Ding, G. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11963–11975, June 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Elharrouss, O., Himeur, Y., Mahmood, Y., Alrabaee, S., Ouamane, A., Bensaali, F., Bechqito, Y., and Chouchane, A. Vits as backbones: Leveraging vision transformers for feature extraction. *Inf. Fusion*, 118:102951, 2025.

Galim, K., Kang, W., Zeng, Y., Koo, H. I., and Lee, K. Parameter-efficient fine-tuning of state space models. In *Forty-second International Conference on Machine Learning*, 2025.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Han, D., Wang, Z., Xia, Z., Han, Y., Pu, Y., Ge, C., Song, J., Song, S., Zheng, B., and Huang, G. Demystify mamba in

vision: A linear attention perspective. *Advances in neural information processing systems*, 37:127181–127203, 2024.

Hatamizadeh, A. and Kautz, J. Mambavision: A hybrid mamba-transformer vision backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25261–25270, 2025.

Hatamizadeh, A., Heinrich, G., Yin, H., Tao, A., Alvarez, J. M., Kautz, J., and Molchanov, P. Fastervit: Fast vision transformers with hierarchical attention. In *International Conference on Learning Representations (ICLR)*, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Khan, S. H., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s):200:1–200:41, 2022.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., and Ren, J. Efficientformer: Vision transformers at mobilenet speed. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12934–12949, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Liu, L., Zhang, M., Yin, J., Liu, T., Ji, W., Piao, Y., and Lu, H. Defmamba: Deformable visual state space model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 8838–8847, 2025.

Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., Jiao, J., and Liu, Y. Vmamba: Visual state space model. In *NeurIPS*, 2024.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Luo, W., Li, Y., Urtasun, R., and Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

NVIDIA. cub::warpscan — cub 2.5 documentation. https://wmaxey.github.io/cccl/cub/api/classcub_1_1WarpScan.html, 2025a. Accessed: 2025-09-15.

NVIDIA. Tensorrt llm. https://github.com/NVIDIA/TensorRT-LLM, 2025b. Accessed: 2025-11-27.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024.

Pei, X., Huang, T., and Xu, C. Efficientvmamba: Atrous selective scan for light weight visual mamba. In *AAAI*, pp. 6443–6451. AAAI Press, 2025.

Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., Wang, W., Banbury, C. R., Ye, C., Akin, B., Aggarwal, V., Zhu, T., Moro, D., and Howard, A. G. Mobilenetv4: Universal models for the mobile ecosystem. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XL*. Springer, 2024.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Doll'ar, P. Designing network design spaces. In *CVPR*, 2020.

Ravi, N., Gabeur, V., Hu, Y., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun,

E., Pan, J., Alwala, K. V., Carion, N., Wu, C., Girshick, R. B., Dollár, P., and Feichtenhofer, C. SAM 2: Segment anything in images and videos. In *ICLR*. OpenReview.net, 2025.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Shaker, A., Wasim, S. T., Khan, S., Gall, J., and Khan, F. S. Groupmamba: Efficient group-based visual state space model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14912–14922, 2025.

Shi, Y., Dong, M., and Xu, C. Multi-scale vmamba: Hierarchy in hierarchy visual state space model. *Advances in Neural Information Processing Systems*, 37:25687–25708, 2024.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., and Bojanowski, P. Dinov3, 2025. URL https://arxiv.org/abs/2508.10104.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *ICML*, volume 139, 2021.

Tschannen, M., Gritsenko, A., Wang, X., Naeem, M. F., Alabdulmohsin, I., Parthasarathy, N., Evans, T., Beyer, L., Xia, Y., Mustafa, B., et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, F., Wang, J., Ren, S., Wei, G., Mei, J., Shao, W., Zhou, Y., Yuille, A., and Xie, C. Mamba-reg: Vision mamba also needs registers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14944–14953, 2025a.

Wang, F., Yang, T., Yu, Y., Ren, S., Wei, G., Wang, A., Shao, W., Zhou, Y., Yuille, A., and Xie, C. Adventurer: Optimizing vision mamba architecture designs for efficiency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30157–30166, June 2025b.

Wang, W., Wang, Z., Suzuki, H., and Kobayashi, Y. Seeing is understanding: Unlocking causal attention into modality-mutual attention for multimodal llms. *CoRR*, abs/2503.02597, 2025c.

Wei, L. H. Mamba ssm cross-platform acceleration: Cuda & metal s6 kernel for jetson and apple silicon. https://github.com/s990093/Mamba-Orin-Nano-Custom-S6-CUDA, 2025. Accessed: 2025-11-27.

Xiao, C., Li, M., Zhang, Z., Meng, D., and Zhang, L. Spatial-mamba: Effective visual state space models via structure-aware state fusion. In *International Conference on Learning Representations (ICLR)*, 2025.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434, 2018.

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., and Gao, J. Focal attention for long-range interactions in vision transformers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 30008–30022, 2021.

Yoshimura, M., Hayashi, T., and Maeda, Y. Mambapeft: Exploring parameter-efficient fine-tuning for mamba. In *The Thirteenth International Conference on Learning Representations*, 2025.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Yu, W. and Wang, X. Mambaout: Do we really need mamba for vision? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4484–4496, June 2025.

Yun, S. and Ro, Y. Shvit: Single-head vision transformer with memory efficient macro design. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Zheng, C. iformer: Integrating convnet and transformer for mobile application. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. In *ICML*. OpenReview.net, 2024.

A. Experimental Setup Details

A.1. Image Classification

We train our SF-Mamba variants (Tiny/Small/Base) on the ImageNet-1K dataset (Deng et al., 2009), which contains 1.28M training images and 50K validation images across 1,000 categories. Following the protocol of MambaVision (Hatamizadeh & Kautz, 2025), we adopt standard data augmentation (RandAugment, Mixup, CutMix) and regularization (Label Smoothing, Stochastic Depth). The detailed hyperparameter settings are summarized in Table 5.

Table 5. Training configurations for SF-Mamba variants on ImageNet-1K. All models are trained for 300 epochs following the MambaVision configuration (Hatamizadeh & Kautz, 2025).

Configuration	**SF-Mamba-T**	**SF-Mamba-S**	**SF-Mamba-B**
Optimizer	LAMB (You et al., 2019)	LAMB	LAMB
Base learning rate	5e-3	5e-3	5e-3
Learning rate schedule	Cosine	Cosine	Cosine
Warmup epochs	20	20	35
Warmup learning rate	1e-6	1e-6	1e-6
Minimum learning rate	5e-6	5e-6	5e-6
Weight decay	0.05	0.05	0.075
Optimizer momentum (β_1)	0.9	0.9	0.9
Optimizer momentum (β_2)	0.999	0.999	0.999
Optimizer epsilon	1e-8	1e-8	1e-8
Gradient clipping (norm)	5.0	5.0	5.0
Total epochs	300	300	300
Batch size (total)	4,096	4,096	4,096
Input resolution	224×224	224×224	224×224
Mixup alpha	0.8	0.8	0.8
CutMix alpha	1.0	1.0	1.0
RandAug (Cubuk et al., 2020)	rand-m9-mstd0.5	rand-m9-mstd0.5	rand-m9-mstd0.5
Label smoothing	0.1	0.1	0.1
DropPath (Huang et al., 2016)	0.2	0.2	0.5
Random erasing prob.	0.25	0.25	0.25
Model EMA	✓	✓	✓
EMA decay	0.9998	0.9998	0.9998
Mixed precision (AMP)	✓	✓	✓

A.2. Object Detection and Instance Segmentation

For MS COCO (Lin et al., 2014), we use Cascade Mask R-CNN (Cai & Vasconcelos, 2019) implemented in MMDetection (Chen et al., 2019). All backbones are initialized from ImageNet-1K pre-training. We use AdamW as the optimizer and adopt the commonly used 3× training schedule. The batch size is 16. Further details follow MambaVision (Hatamizadeh & Kautz, 2025).

A.3. Semantic Segmentation

For ADE20K (Zhou et al., 2017), we use UperNet (Xiao et al., 2018) implemented in MMSegmentation (Contributors, 2020). Backbones are initialized with ImageNet-1K pre-training. We use AdamW as the optimizer with batch size 16. A polynomial learning rate decay schedule is applied, consistent with MambaVision (Hatamizadeh & Kautz, 2025).

A.4. Throughput Measurement

We measure throughput on an NVIDIA A100 40GB GPU with a batch size of 128 and input images of size 224×224 using automatic mixed precision, following established protocols (Hatamizadeh et al., 2024; Hatamizadeh & Kautz, 2025). Note that some models in Fig. 1 have non-224×224 input resolution following the original setup (e.g. MobileNetV4-H-M (Qin

et al., 2024): 256×256, MobileNetV4-H-L: 384×384, EfficientViT-B2(r256) (Cai et al., 2023): 256×256, EfficientViT-B3(r288): 288×288, SHViT (Yun & Ro, 2024): 384×384). Our software environment consists of CUDA 12.4, cuDNN 9, and PyTorch 2.6.0. To ensure a fair comparison, we measure the throughput of all previous methods under the same experimental settings. We report the speed of the faster memory format between channel last and channel first. The reported throughput values are the median over 500 inference runs, and for ours-T, the variation across 10 trials was 7600 ± 11.

B. Implementation Details

B.1. Macro-Architecture

MambaVision (Hatamizadeh & Kautz, 2025) combining Attention and Mamba, is a state-of-the-art macro-level architecture for vision tasks, excelling in speed, performance, and scalability. Therefore, our macro-architecture follows MambaVision with a four-stage hierarchical design. Given an input image $I \in \mathbb{R}^{H\times W\times 3}$, the stem and successive stages transform the resolution and channel dimension as follows:

$$\begin{aligned}
&\text{Stage 1:} && I \xrightarrow{\text{Stem + ConvBlock}\times N_1} \tfrac{H}{4}\times\tfrac{W}{4}\times D,\\
&\text{Stage 2:} && \tfrac{H}{4}\times\tfrac{W}{4}\times D \xrightarrow{\text{Downsample + ConvBlock}\times N_2} \tfrac{H}{8}\times\tfrac{W}{8}\times 2D,\\
&\text{Stage 3:} && \tfrac{H}{8}\times\tfrac{W}{8}\times 2D \xrightarrow{\text{Downsample + }\textit{MambaBlock}\times N_3/2\text{ + }\textit{AttenBlock}\times N_3/2} \tfrac{H}{16}\times\tfrac{W}{16}\times 4D,\\
&\text{Stage 4:} && \tfrac{H}{16}\times\tfrac{W}{16}\times 4D \xrightarrow{\text{Downsample + }\textit{MambaBlock}\times N_4/2\text{ + }\textit{AttenBlock}\times N_4/2} \tfrac{H}{32}\times\tfrac{W}{32}\times 8D,\\
&\text{Classifier:} && \tfrac{H}{32}\times\tfrac{W}{32}\times 8D \xrightarrow{\text{Global AvgPool + Linear}} \mathbb{R}^{\#\text{classes}}.
\end{aligned} \tag{6}$$

where N_i denotes the number of blocks to apply sequentially. In Stage 3 and 4, $N_i/2$ Mamba blocks are applied followed by $N_i/2$ Attention blocks. The Mamba Block consists of a MambaVision Mixer and an MLP. The MambaVision Mixer takes a patch sequence as input $X_{\text{in}} \in \mathbb{R}^{B\times T\times D}$ and processes it through two parallel branches: a selective SSM branch and a local convolutional branch. Formally,

$$\begin{aligned}
X_1 &= \text{SSM}\Big(\sigma(\text{Conv}(\text{Linear}_{D\to D}(X_{\text{in}})))\Big), \quad X_2 = \sigma(\text{Conv}(\text{Linear}_{D\to D}(X_{\text{in}}))),\\
Y &= \text{Linear}_{2D\to D}(\text{Concat}(X_1, X_2)).
\end{aligned} \tag{7}$$

Here, σ is a SiLU activation (Elfwing et al., 2018), Conv is a 1-D depthwise convolution, and $\text{SSM}(\cdot)$ denotes the SSM with selective scan $h_t = A_t h_{t-1} + B_t x_t$, $y_t = C_t h_t$. The two paths are fused and projected back to dimension D, yielding the output Y. Unlike many visual Mamba methods (Liu et al., 2024; Wang et al., 2025a), the MambaVision Mixer accelerates processing by adopting a simple unidirectional scan. However, due to the *causality constraint*, it cannot access future patches from past positions, so future-to-past information flow relies on subsequent Attention blocks.

B.2. Implementation Optimization for Faster Inference

As indicated by "impl. opt" in Tab. 2, we apply several implementation level optimizations not written in the method section to accelerate inference. The details of these implementation optimizations are listed below:

- **Removal of unused row-dimension chunking in the Mamba SSM kernel**: As with VMamba (Liu et al., 2024), we remove the unused row (channel) dimension chunking feature from the Mamba SSM kernel. This allows more intermediate variables to be handled as float values rather than float arrays, resulting in improved speed.
- **Suppressing hidden state output during inference**: The Mamba SSM kernel is modified so that it does not output hidden states except during training. Since hidden states are only needed for backpropagation, avoiding their output during inference reduces unnecessary memory write.
- **Replacing linear layers with pointwise 1D Convolution**: As with VMamba, we replace the linear layer that output the Δt tensor with pointwise 1D convolutions. This reduces unnecessary tensor rearrangement.
- **Auxiliary token swapping with a Triton CUDA kernel**: Although the computational cost of the swapping is not significant, swapping data at non-contiguous positions is needed, especially with the batch folded data. Converting this process to a Triton CUDA kernel improves throughput slightly by about 40 img/s for ours-T and about 10 img/s for ours-B, although we can use none-Triton swapping for simplicity.

B.3. Segmentation and Object Detection

In image classification, we followed the MambaVision meta-architecture. However, for object detection and instance segmentation on the COCO dataset (Lin et al., 2014), and semantic segmentation on ADE20K (Zhou et al., 2017), we make some modifications. The reason is that processing high-resolution images with Attention incurs significant computational cost. Based on our analysis, it appears that MambaVision mistakenly omits the computational cost of Attention in terms of FLOPs for the COCO and ADE20K tasks. Therefore, the FLOPs values for MambaVision in our table differ from those reported in the original paper.

To address this, we made two improvements to create a more lightweight model architecture. The first is to remove excessive padding regions. In MambaVision, large padding areas are added in both the Mamba Block and the Attention Block to serve as additional computation regions, thereby improving accuracy. Although it leads to a degradation in accuracy, we reduce computational cost by removing these extra padding regions and lowering the resolution in Stage 3 and Stage 4 (e.g. Stage 3: 112×112 to 84×84, Stage 4: 56×56 to 42×42 for COCO).

The second improvement is the use of windowed Attention. Since Attention has a quadratic cost with respect to token length, we reduce computational cost by applying local windowed Attention to Stage 3, which has a long sequence length. This also results in a slight drop in performance.

After applying these changes, our model architecture is as follows: The stem layer, Stage 1, and Stage 2 are convolution-based and process the input image directly. Stage 3 processes features padded to a resolution of 84×84 for COCO and 64×64 for ADE20K. Stage 4 processes images at 42×42 for COCO and 32×32 for ADE20K. Padding is necessary because these tasks require handling images with various aspect ratios and resolutions. For task-specific decoders–Cascade Mask R-CNN (Cai & Vasconcelos, 2019) (for COCO) and UperNet (Xiao et al., 2018) (for ADE20K)–the padding regions are removed before input. When using windowed Attention, the window size in Stage 3 is set to 42×42 for COCO and 32×32 for ADE20K. During training on ADE20K, the model is trained with an input resolution of 512×512, whereas during evaluation it needs to process resolutions up to 2048×512. Therefore, in Stage 3 and Stage 4 during the evaluation, both the Mamba and Attention blocks handle feature maps larger than 64×64 or 32×32 by dividing them into windowed patches for processing.

B.4. Other Details

When layers are randomly dropped, auxiliary tokens may or may not be swapped, leading to inconsistency. To address this, we slightly modify DropPath (Huang et al., 2016) for auxiliary token swapping: while other tokens can be dropped, auxiliary tokens are always preserved and never dropped. When the DropPath rate is low, as in ours-T, this modification does not improve performance. However, when the DropPath rate is high, as in ours-B, it improves performance by about 0.2 points.

C. Additional Experiments

C.1. Preliminary Evaluation on Multi-directional Scan Cost

We measure how much existing multi-directional scan methods affect throughput. As representative examples of multi-directional scan, we experiment with bi-directional scan (Zhu et al., 2024) and cross-scan (Liu et al., 2024), which are commonly used as the basis for many scanning methods (Wang et al., 2025a; Shi et al., 2024; Pei et al., 2025). To accurately identify the causes of performance degradation, we conduct the following three simple experiments. The first experiment measures the throughput using the original model structure as proposed, which includes multi-directional scan. The second experiment measures the throughput of a model whose scan directions are replaced with forward-only scans. The last experiment measures the throughput of a model where all non-forward scan directions are removed from the original model. The difference between the first and second experiments reflects the time spent on reordering tokens, which is required by multi-directional scans. The difference between the second and third experiments indicates the time cost of performing scans in parallel.

Fig. 6 shows how much of the total inference time is occupied by these components. Surprisingly, we found that token reordering, which is not reflected in FLOPs, accounts for 5–8% of the total processing time in models using multi-directional scan. Furthermore, performing parallel multi-directional scans consumes an additional 28–42% of processing time, which means that the accuracy gains of multi-directional scan must outweigh this cost. In the case of VMamba, the time spent rearranging the data between 2D and 1D formats is additionally hidden under the ”others” category.

Our method is also included in the table as a reference. Direct comparison is difficult since our model uses Attention too and the proportion of Mamba blocks is relatively small. However, auxiliary token swapping in our method results in negligible processing time. As a result, in addition to the effectiveness of batch folding, our model is significantly faster, although all three models have nearly identical FLOPs.

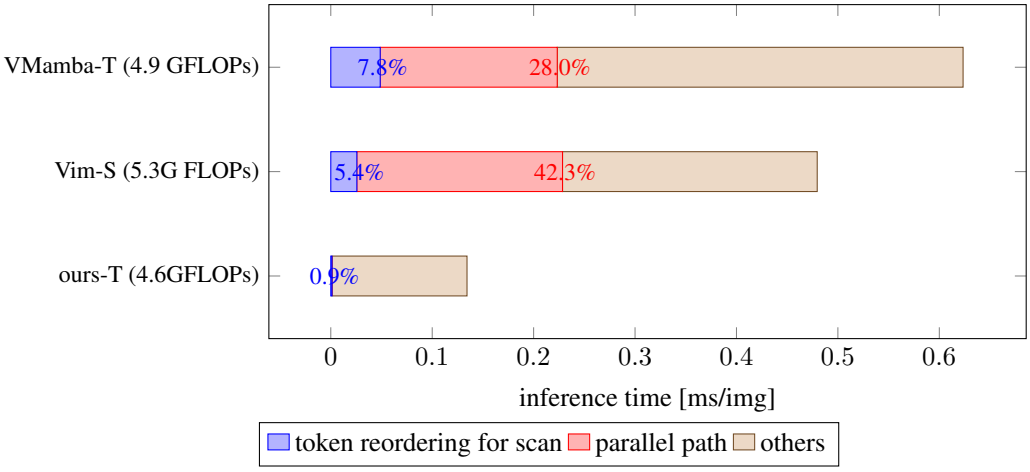


Figure 6. Computational cost of multi-directional scan. This includes the time required to reorder tokens for scanning from multiple directions, and the additional processing time incurred by setting up parallel paths.

C.2. Effective Receptive Field Analysis

To better understand how our model captures spatial dependencies, we conduct an Effective Receptive Field (ERF) analysis (Luo et al., 2016; Ding et al., 2022) following the methodology of Liu et al. (2024). The ERF is computed by measuring the squared gradient of the output features with respect to the center pixel, which highlights the regions most influential for each prediction. Fig. 7 shows the ERF corresponding to the layers up to the Stage 3 Mamba blocks of tiny sized models. So, the ERF for only Convolution and two Mamba blocks is shown. MambaVision uses a simple unidirectional scan, which prevents it from accessing future tokens (i.e., the lower part of the image) beyond what can be captured by convolution. In contrast, SF-Mamba leverages auxiliary token swapping, allowing it to account for both past and future tokens with similar strength. Since the information propagation of auxiliary token swapping follows the mechanism of SSM, it can be effectively achieved with only two tokens. Fig. 8 compares the ERFs of entire models. SF-Mamba leverages auxiliary patch swapping to facilitate a global receptive field while maintaining high throughput. Unlike attention-based architectures whose cost scales quadratically with the sequence length, SF-Mamba avoids this overhead thanks to its state-space formulation. This demonstrates that SF-Mamba achieves global context modeling with improved computational efficiency.

C.3. Detailed Evaluation in Semantic Segmentation

Table 6. Semantic segmentation performance on ADE20K dataset using UperNet. We compare with Swin Transformer (Liu et al., 2021), Focal Transformer (Yang et al., 2021), and MambaVision (Hatamizadeh & Kautz, 2025). All models are trained at a resolution of 512×512 while FLOPs are calculated with an input size of 2048×512. Frames per second (FPS) are measured with a batch size of 1. SF-Mamba♣ uses a windowed attention to save computational cost.

	Tiny-size				Small-size				Base-size			
Backbone	Para	FLOPs	mIoU	fps	Para	FLOPs	mIoU	fps	Para	FLOPs	mIoU	fps
Swin	60M	945G	44.5	40.0	81M	1038G	47.6	25.7	121M	1188G	48.1	25.4
Focal	62M	998G	45.8	38.9	85M	1130G	48.0	24.0	126M	1354G	49.0	23.4
MambaVision	62M	1085G	46.0	45.0	81M	1166G	48.2	40.9	130M	1520G	49.1	37.3
SF-Mamba	62M	1085G	**47.2**	47.9	81M	1166G	**48.5**	45.4	130M	1520G	**50.1**	42.6
SF-Mamba♣	62M	950G	46.5	**48.7**	81M	1014G	48.1	**47.3**	130M	1180G	49.1	**42.7**

Tab. 6 presents the detailed evaluation results for semantic segmentation shown in 5. Comparing SF-Mamba and SF-Mamba♣, the use of window attention significantly reduces the FLOPs of SF-Mamba♣. The inference speed, however,

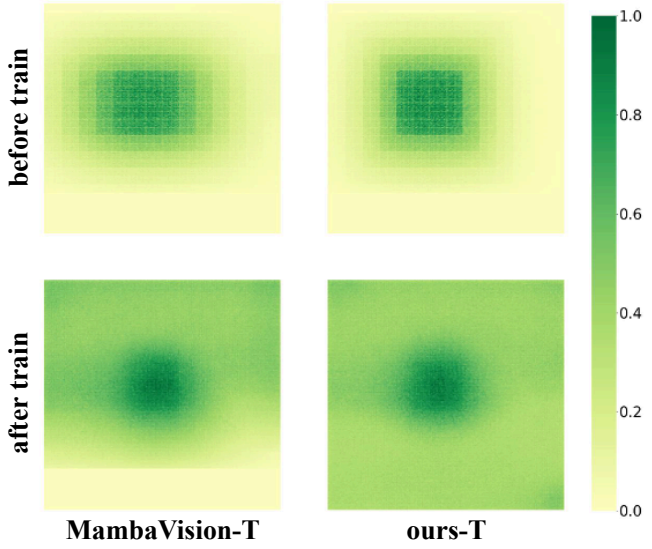


Figure 7. Effective Receptive Field (ERF) comparison. This ERF corresponds to the layers up to the Stage 3 Mamba blocks. MambaVision uses a simple unidirectional scan, which prevents it from accessing future tokens (i.e., the lower part of the image) beyond what can be captured by convolution. In contrast, SF-Mamba leverages auxiliary token swapping, allowing it to account for both past and future tokens with similar strength. Since the auxiliary token swapping information propagation follows the mechanism of SSM, it can be effectively achieved with just two tokens.

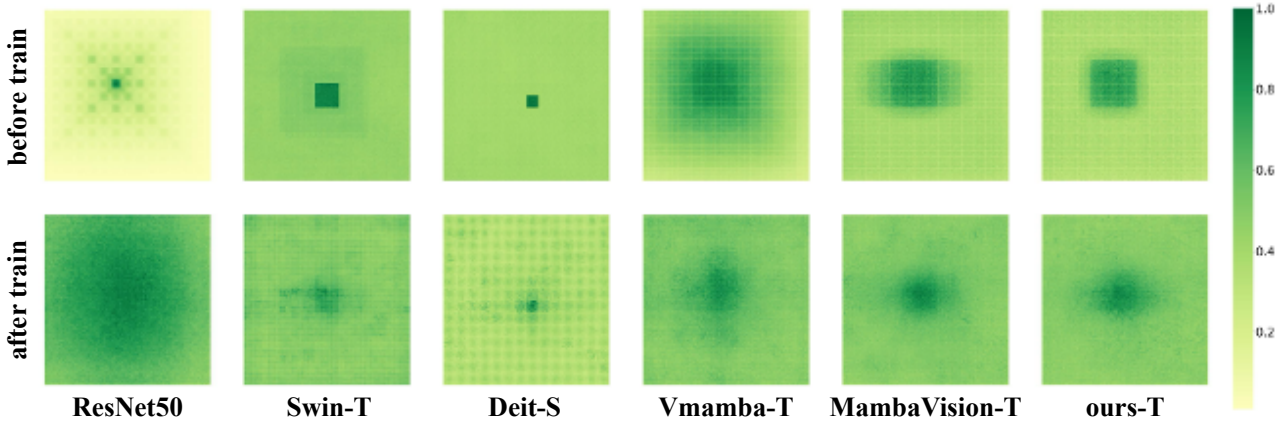


Figure 8. Effective Receptive Field (ERF) comparison of the entire model. SF-Mamba achieves globally distributed ERFs with reduced computational complexity.

does not change much because even with long token sequences, FlashAttention2 provides considerable acceleration, which offsets the additional time required for reshaping operations introduced by window attention. On older GPUs such as the V100, where FlashAttention2 is not available, SF-Mamba♣ runs faster.

C.4. Object Detection and Instance Segmentation

Table 7. Object detection and instance segmentation performance on MS COCO dataset using Cascade Mask R-CNN. All models are trained with 3× schedule at 1280×800 resolution. We compare with ConvNeXt (Liu et al., 2022), Swin Transformer (Liu et al., 2021), and MambaVision (Hatamizadeh & Kautz, 2025). SF-Mamba♣ uses a windowed Attention (no window for Mamba blocks) to save computational cost.

Backbone	Params	FLOPs	fps	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Swin-T	86M	745G	26.3	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-T	86M	741G	**32.1**	50.4	69.1	54.8	43.7	66.5	47.3
MambaVision-T	89M	1118G	19.4	**51.1**	70.0	55.6	**44.3**	67.3	47.9
SF-Mamba-T	89M	741G	27.8	51.0	69.9	55.3	44.2	67.1	48.0
SF-Mamba♣-T	89M	**659G**	28.3	50.9	69.9	55.0	44.1	66.9	47.7
Swin-S	107M	838G	18.7	51.9	70.7	56.3	45.0	68.2	48.8
ConvNeXt-S	108M	827G	28.0	51.9	70.8	56.5	45.0	68.4	49.1
MambaVision-S	107M	1192G	20.5	52.3	71.1	56.7	45.2	68.5	48.9
SF-Mamba-S	107M	817G	28.7	**52.4**	71.1	56.7	**45.4**	68.5	49.1
SF-Mamba♣-S	107M	**731G**	**28.8**	52.1	71.0	56.4	45.2	68.4	48.8
Swin-B	145M	982G	18.6	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-B	146M	**964G**	26.0	52.7	71.3	57.2	45.6	68.9	49.5
MambaVision-B	155M	3000G	16.4	**52.8**	71.3	57.2	45.7	68.7	49.4
SF-Mamba-B	155M	1185G	26.8	**52.8**	71.3	57.2	**45.8**	68.9	49.4
SF-Mamba♣-B	155M	992G	**27.6**	52.6	71.3	57.2	45.7	69.0	49.2

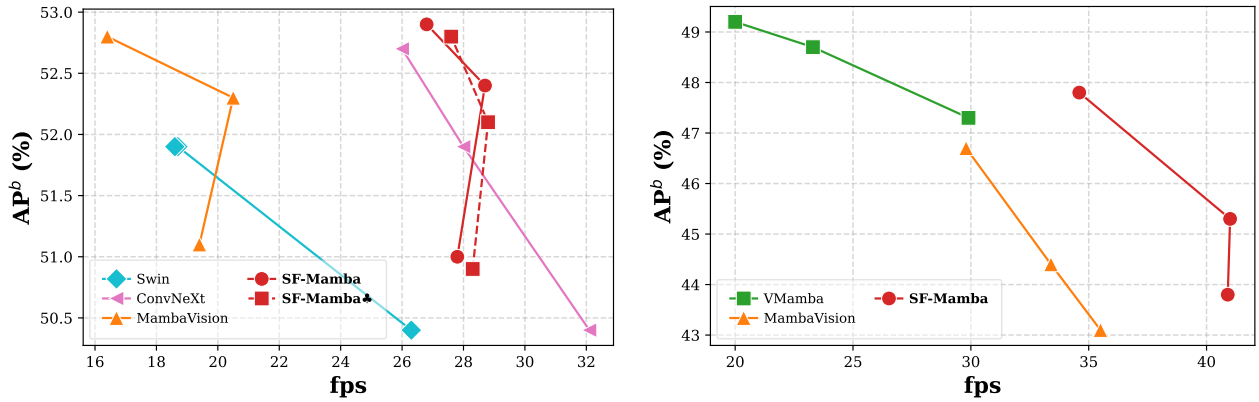
Experimental Setup. We evaluate on MS COCO 2017 (Lin et al., 2014) using Cascade Mask R-CNN (Cai & Vasconcelos, 2019) as the detection framework. The task involves localizing objects with bounding boxes (detection) and predicting pixel-level masks for each instance (segmentation). We follow the standard 3× training schedule. We report both bounding-box average precision (AP) and mask AP metrics following the COCO evaluation protocol (Lin et al., 2014).

Results. Tab. 7 presents the object detection and instance segmentation results on MS COCO. Our approach again achieves improvements in both accuracy and efficiency over the baseline and also outperforms the Swin and Focal Transformer, indicating its general applicability. As discussed in Appendix C.6, by removing the extensive padding region used in the baseline and replacing global attention with window-based attention, we achieve substantial efficiency gains at the expense of some accuracy. For example, SF-Mamba♣-S, utilizing window Attention to save computational cost, has smaller FLOPs than the tiny size models while achieving 1.0 and 0.9 points higher AP^b and AP^m. Thanks to the introduction of state swapping, we attain performance comparable to or even surpassing the baseline. The clear improvement over the existing methods can be seen in Fig. 9(a).

C.5. Object Detection and Instance Segmentation with Other Detection Heads

Experimental Setup. To evaluate the generality of our method on downstream tasks and compare it with other existing image encoders, we also perform experiments on the COCO dataset (Lin et al., 2014) using Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) detection heads. Following prior work (Radosavovic et al., 2020; Liu et al., 2024), we train all models with the 1× schedule (12 epochs). The baselines we compare are RegNetX (Radosavovic et al., 2020), VMamba (Liu et al., 2024), and MambaVision (Hatamizadeh & Kautz, 2025). For MambaVision, we disable excessive padding and instead use the same minimal padding strategy as our method to compare with a comparable computational cost.

Results. We first present the Faster R-CNN results in Table 8. Under the aligned experimental settings and matched computational cost, our method achieves a substantial performance improvement over MambaVision.



(a) (b)

Figure 9. Accuracy-speed trade-off on MS COCO using (a) Casecade Mask R-CNN (Cai & Vasconcelos, 2019) and (b) Mask R-CNN (He et al., 2017). Casecade Mask R-CNN is trained with the 3x schedule while Mask R-CNN is trained with the 1x schedule. In MambaVision and in our model built on its macro-architecture, the tiny and small variants show a reversal in speed. This is because the tiny model has three Attention layers in stage 3 and two in stage 4, whereas the small model has two Attention layers in stage 3 and three in stage 4. When high-resolution images are used as input, Attention computation becomes the bottleneck. Consequently, even though the tiny model has fewer parameters, its throughput becomes lower due to the larger number of Attention layers in stage 3.

The results with Mask R-CNN is shown in Table 9 and Fig. 9, showing consistent accuracy and speed improvements over our baseline again. Although our method is weaker in accuracy compared to VMamba within the same categories (T, S, or B), when comparing VMamba-**T** and SF-Mamba-**B**, SF-Mamba-B surpasses VMamba-T in both speed and accuracy, clearly demonstrating a superior performance–throughput trade-off. The Figure 9(b) should be easy to understand the trade-off.

Table 8. Detection on COCO dataset using Faster R-CNN. The models are trained with 1x schedule (12 epochs).

Backbone	AP box	fps
RegNetX-3.2GF	39.9	31.6
MambaVision-T	42.4	37.2
SF-Mamba-T	**43.2**	**41.7**
MambaVision-S	43.9	37.2
SF-Mamba-S	**44.9**	**40.8**
MambaVision-B	46.2	30.0
SF-Mamba-B	**47.6**	**34.8**

C.6. Evaluation of Excessive Padding and Windowed Attention in Segmentation and Detection Tasks

As outlined in Appendix B.3, Tab. 10 presents an ablation study on the impact of excessive padding regions and the use of windowed Attention in terms of computational cost and accuracy. Introducing excessive padding enables the model to utilize additional spatial regions for additional computational area, which leads to a modest accuracy gain. However, due to the quadratic scaling of Attention with respect to token length, this improvement comes at a substantial computational cost.

We further examine the effect of applying Attention or Mamba within local windows. This consistently resulted in accuracy degradation, suggesting that both mechanisms are effective in capturing long-range dependencies. It is an advantage over convolutional approaches. Despite the drop in accuracy, windowed Attention significantly reduces computational overhead. In contrast, Mamba maintains linear complexity with respect to sequence length, which means that windowing does not reduce its computational cost. Based on these findings, our SF-Mamba♣ applies windowing exclusively to Attention, while utilizing Mamba for global modeling. Since windowed Attention restricts complete future-to-past token information flow, our auxiliary token swapping mechanism plays a critical role in enabling bidirectional context propagation. For high-resolution inputs, the benefits of Mamba over Attention become even more pronounced. This indicates that increasing the use of Mamba may further enhance performance in high-resolution segmentation and detection tasks.

Table 9. Detection and instance segmentation on COCO dataset using Mask R-CNN. The models are trained with 1x schedule (12 epochs).

Backbone	AP_{box}	AP_{mask}	FPS
VMamba-T	47.3	42.7	29.9
MambaVision-T	43.1	40.0	35.5
SF-Mamba-T	43.8	40.3	40.9
VMamba-S	48.7	43.7	23.3
MambaVision-S	44.4	41.0	33.4
SF-Mamba-S	45.3	41.5	41.0
VMamba-B	49.2	44.1	20.0
MambaVision-B	46.7	42.8	29.8
SF-Mamba-B	47.8	43.5	34.6

Table 10. The impact of excessive padding regions and the use of windowed Attention in terms of computational cost and accuracy. With the excessive padding setting, Stage 3 uses large padding sizes — 112×112 for COCO and 64×64 for ADE20K. In contrast, the *w/o pad* configuration minimizes padding to match the training image sizes, resulting in 84×84 for COCO and 32×32 for ADE20K. Regarding local windows: A3 refers to the windowed Attention in Stage 3 and M3 refers to the windowed Mamba in Stage 3. The configurations used for our models, SF-Mamba and SF-Mamba♣, are highlighted. For ADE20K, since large images are processed during testing, we retain the large padding.

		number of window				ADE20K		COCO		
arch.	w/o pad	A3	A4	M3	M4	FLOPs	mIoU	FLOPs	mAP^b	mAP^m
w/o swap						1085G	46.0	1118G	50.9	44.1
w/ swap						1085G	47.2	1118G	51.2	44.5
w/ swap	✓					950G	45.8	741G	51.0	44.2
w/ swap		4				950G	46.2	741G	50.9	44.0
w/ swap	✓	4				942G	45.6	736G	50.9	44.0
w/ swap	✓	4	4			941G	45.6	649G	50.5	44.0
w/ swap	✓	4	4	4	4	941G	45.4	649G	50.3	43.9

C.7. Effect of the Number of Auxiliary Tokens

We conduct an additional experiment by varying the number of auxiliary tokens. In addition to the proposed paired token swapping, we include a single-token setting ($T = 1$), where the auxiliary token is placed at the end of the sequence in odd-numbered Mamba blocks and relocated to the beginning in even-numbered blocks.

Table 11 summarizes the results on ImageNet-1K. We observe no performance gain when increasing from $T = 0$ (uni-scan) to $T = 1$. However, there is a clear improvement from $T = 1$ to $T = 2$, while performance is saturated for $T > 2$. This suggests that the global information required for ImageNet classification is relatively compact, and that two auxiliary tokens provide sufficient degrees of freedom to capture the dominant global components (e.g., lighting conditions and scene-level semantics) without overloading the SSM capacity.

Furthermore, the qualitative difference between the $T = 1$ and $T = 2$ settings provides insight into the underlying mechanism. In the $T = 1$ case, although the auxiliary token can be relocated across layers, it must simultaneously serve two roles: (1) carrying global information to the next layer, and (2) enabling selective readout for choosing the global feature in the next layer. This dual responsibility entangles the representation and limits effective feature selection.

In contrast, the $T = 2$ swapping design allows the two tokens to specialize. The token placed at the beginning of the sequence injects global contextual information into the SSM, while the token at the end selectively reads out information for the subsequent layer. This separation of roles facilitates more effective global information processing.

These results confirm that the proposed swapping mechanism operates as intended, and that a minimal configuration of two auxiliary tokens is sufficient to improve performance.

Table 11. Ablation study about the number of auxiliary tokens on ImageNet-1K.

# auxiliary tokens	Acc. [%]
0	82.2
1 (relocation)	82.2
2 (swap)	**82.5**
4 (swap)	**82.5**
6 (swap)	82.3

C.8. Comparison with Linear Attention

Table 12. Comparison with linear attention on ImageNet-1K.

macro arch.	layer	Acc. [%]	throughput
MambaVision-T	MILA Linear Attention	82.4	5504
MambaVision-T	Simple Linear Attention	82.2	7437
MambaVision-T	Mamba with swap (ours)	**82.5**	**7600**
MILA-T	MILA Linear Attention	**83.5**	1811
MambaVision-S	Mamba with swap (ours)	**83.5**	**5639**

We further investigate the effectiveness of Mamba with auxiliary token swapping by comparing it with linear attention-based alternatives. In particular, we consider MILA (Han et al., 2024), which improves accuracy with a 2D-specific linear attention. Following the same philosophy as our other ablation studies, we replace the Mamba blocks in MambaVision with linear attention modules. Specifically, we evaluate two variants: the original MILA linear attention and a simplified linear attention where 2D-specific components (i.e., 2D-RoPE and LePE) are removed.

Table 12 shows the comparison on ImageNet-1K. When replacing the SSM part in the Mamba block with MILA-style linear attention, performance is slightly degraded, while throughput is also lower. The simplified linear attention improves throughput but still underperforms our method. In contrast, our approach achieves both higher accuracy and faster inference.

We also observe that mixed-precision training for linear attention is unstable, requiring 32-bit floating-point training, consistent with the original MILA paper. This significantly increases the training cost. In contrast, our method remains stable under mixed precision, making it more efficient in practice.

C.9. Applicability to other Vision Mamba Variants

Our two core contributions—auxiliary patch swapping and batch folding with periodic reset—can be integrated into other visual Mamba variants. To demonstrate this, we add experiments on Vim (Zhu et al., 2024) architecture as shown in Table 13.

When we remove the modules responsible for the inverse-direction scan in the Vim-S architecture and make the model uni-directional, the accuracy drops, but the speed improves significantly. By introducing the proposed auxiliary token swapping, we can recover most of the lost accuracy while preserving the improved speed. Furthermore, unlike the MambaVision macro-architecture, Vim makes extensive use of Mamba blocks, which allows our batch folding to yield a substantial speed improvement. The resulting model achieves a speed similar to Vim-T but with substantially better accuracy than Vim-T, demonstrating that our method offers a clearly superior accuracy–throughput trade-off.

Our method has proven effective for architectures such as MambaVision and Vim but, there are also limitations on which Visual Mamba models it can be applied to. For example, it is difficult to adapt our approach to architectures like VMamba (Liu et al., 2024), which incorporate 2D convolutions inside the Mamba module. This is because these models must convert the data back into a 2D format at every layer, but once auxiliary tokens are added, the data no longer conform to the original 2D format. In addition, they cannot process data in the batch-folded representation; instead, they must reconvert the data into the non-batch-folded format at every layer, which degrades the speed-up.

C.10. Throughput Evaluation Under Various Scenarios

Here, we evaluate throughput across a variety of scenarios.

Table 13. Comparison on Vim-S macro-architecture. To match the parameter count, we increase the channel dimension from 384 to 400 in the uni-scan model. By applying our method to Vim-S, we can significantly improve its performance over Vim-T while maintaining inference speed comparable to Vim-T.

size	scan	Params	MACs	img/s	acc.
S	parallel-bi scan (Vim)	26M	5.3G	1079	**80.3**
S	uni-scan	26M	4.9G	1639	79.3
S	uni-scan + swap (ours)	26M	5.0G	1614	80.1
S	uni-scan + swap (ours) + Bfold (ours)	26M	5.0G	**2022**	80.1
T	parallel-bi scan (Vim)	7M	1.5G	2094	76.3

Higher Input Resolutions. A throughput comparison at higher input resolutions is shown in 14. These results show that our proposed improvements preserve their benefits even as resolution scales. The strategy using windowed Attention while using Mamba globally (SF-Mamba♣) remains particularly robust under extremely large inputs. This may reduce training and inference costs in high-resolution applications such as medical imaging, aerial monitoring, and robotics.

Table 14. Throughput (images/s) for different models and resolutions. OOM denotes out-of-memory with an A100 40GB GPU. We use a windowed Attention with a 32×32 size for SF-Mamba♣, the same as the ADE20K setup. The "-" for SF-Mamba♣ means that the feature sizes of both stage 3 and 4 are less than 32×32, so the same with SF-Mamba.

Model (batch size)	224	448	896	1792	3584
VMamba-T (bs=32)	1384	402	107	5	OOM
FasterViT-0 (bs=32)	1415	1400	418	99	OOM
MambaVision-T (bs=32)	3770	1578	324	50	5
SF-Mamba-T (bs=32)	**3962**	**1777**	397	61	6
SF-Mamba♣-T (bs=32)	-	-	**427**	**105**	**27**
VMamba-T (bs=1)	62	62	62	24	5
FasterViT-0 (bs=1)	44	43	43	43	12
MambaVision-T (bs=1)	119	121	120	48	5
SF-Mamba-T (bs=1)	**126**	**126**	**125**	54	6
SF-Mamba♣-T (bs=1)	-	-	120	**89**	**26**

Different Batch Sizes. The throughput measured under different batch sizes is summarized in Table 15. Although auxiliary token swapping introduces a slight increase in computational cost, the results show that throughput consistently improves thanks to batch folding and other optimizations.

Table 15. Throughput with various batch sizes

Arch.	1	32	128	256	512	1024
MambaVision-T	119	3770	6662	7025	7134	7271
ours-T	**126**	**3962**	**7600**	**7801**	**8009**	**8190**
MambaVision-B	96	2798	2974	3128	3176	3206
ours-B	**98**	**3168**	**3534**	**3592**	**3641**	**3685**

C.11. Contribution of Attention and Mamba

Here, we analyze the contribution of Attention and Mamba as shown in Table 16. These results show that while Attention provides beneficial bidirectional information flow, it alone is not sufficient to match the full hybrid model. In contrast, SSM alone lags behind, but incorporating the swapping mechanism yields a clear improvement. The best performance is achieved only when both components–Attention and SSM (with swap)–are present. This supports our claim that token swapping plays a complementary role to Attention rather than replacing it. Thanks to our batch folding with periodic reset and auxiliary-token swapping, we can leverage Mamba to achieve improvements in the accuracy–speed trade-off even for

low-resolution inputs.

Table 16. The contribution of Attention and Mamba.

Arch.	**Params**	**img/s**	**acc.**
Attention only	34.2M	**7803**	82.3%
SSM only	29.4M	6238	80.2%
SSM only (w/ our Bfold and swap)	29.4M	7306	81.0%
Hybrid	31.8M	6979	82.2%
Hybrid (w/ our Bfold and swap)	31.8M	7600	**82.5%**