

PVMamba: Parallelizing Vision Mamba via Dynamic State Aggregation

Fei Xie¹ Zhongdao Wang² Weijia Zhang¹ Chao Ma^{1,*}

¹ MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

² Huawei Noah's Ark Lab

{jaffe031, weijia.zhang, chaoma}@sjtu.edu.cn wangzhongdao@huawei.com

Abstract

Mamba, an architecture with RNN-like sequence modeling of State Space Model (SSM), has demonstrated promising capabilities in long-range modeling with high efficiency. However, *Mamba* models struggle with structured 2D visual data using sequential computing, thereby lagging behind their attention-based counterparts. In this paper, we propose a Parallel Vision Mamba (PVMamba), a novel SSM architecture tailored for visual data. PVMamba encompasses two key designs: 1) Based on the sparsity and adjacency of visual signals, we parallelize the sequential computing through three core steps, termed Dynamic State Aggregation (DSA), i.e., parallelization, alignment, and aggregation. DSA generates the hidden state in SSM by a feasible spatial aggregation, thereby overcoming the inherent sequential constraints. 2) Along with maintaining linear computational complexity, we apply a dynamic operator to learn the spatial samplings for each hidden state. To further boost the local modeling capability, we restrict the dynamic operator to the neighboring pixels in shallow layers. We also devise a layer multiplexing technique to stabilize the training and reduce the learning redundancy. PVMamba is a versatile backbone network with dynamic operators for various vision tasks, such as image classification and dense prediction. Extensive experiments show that PVMamba achieves state-of-the-art performance on a range of benchmarks. The code is available at <https://github.com/VISION-SJTU/PVMamba>.

1. Introduction

The architecture of Structured State Space Models (SSMs) [8, 9, 13, 26, 38] has become increasingly popular due to their remarkable computational efficiency in long sequence modeling. Recently, there has been a growing trend in applying Mamba [25, 31, 62, 71], an improved SSM variant, as the core module for various vision tasks, such as im-

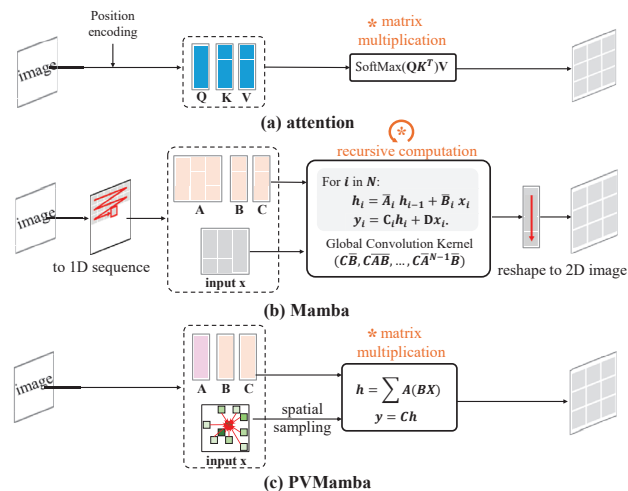


Figure 1. (a) shows attention [33, 49, 65], whose computational costs are expensive, while it is implemented by hardware-friendly matrix operations. (b) is Mamba scheme [22, 31, 71] with linear complexity while suffering from recursive computation and sequential transformation. (c) Our PVMamba aims to preserve the merits of complexity and break the sequential constraints.

age classification [31, 71], dense prediction [28, 36, 41, 62], and generative tasks [7, 14, 20, 47]. These applications show the potential of Mamba in the visual domain.

Despite their advances in model efficiency, SSMs, which originated from sequential language tasks [54], lack inherent adaptation for the visual domain. This limitation makes many existing SSM architectures lag behind their transformer [5, 33, 49] counterparts in performance on vision understanding tasks. We carefully examine the merits and demerits of the visual operators as illustrated in Fig. 1. The transformer can flexibly attend to the pixels in 2D regions and be efficient in matrix calculation while suffering from quadratic computational costs. In contrast, Mamba benefits from linear complexity and processes each pixel in sequence, maintaining awareness of their relative order. However, the sequential nature [8, 13] in Mamba introduces two main constraints when handling 2D visual data: 1) Vision Mamba needs to convert 2D visual data into 1D sequential

*Corresponding author.

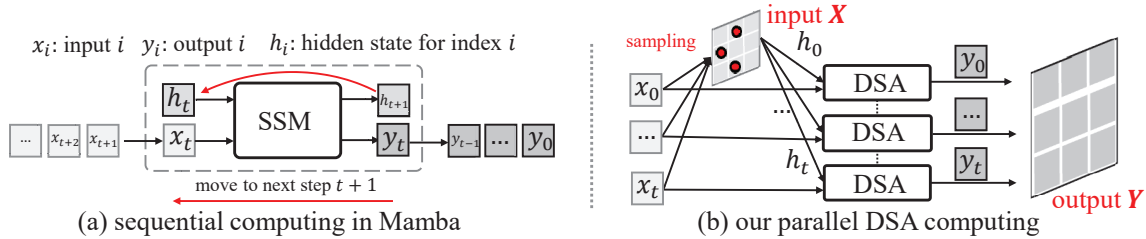


Figure 2. (a) the sequential pipeline of hidden state computing in Mamba, which conditions on previous state and input feature; (b) DSA parallelizes the hidden state computation through a feasible spatial aggregation from the input feature.

formats, which limits its spatial modeling capabilities. 2) A recursive pipeline brings latency, which is unfriendly to the current parallel deep learning paradigm [39]. While the two mentioned sequential constraints are partially addressed by developing more complex scanning strategies [22, 31] and hardware-aware optimizations [8], the core operator of vision Mamba still follows a sequential pipeline, as shown in Fig. 2(a). Thus, it raises a natural question: *Can we parallelize the Mamba operator to break the sequential constraints while preserving the merits of the Mamba?*

In this work, we propose a novel PVMamba scheme, which parallelizes the Mamba scheme via three core steps: *parallelization*, *alignment*, and *aggregation*. Our intuitive idea comes from the fact that the visual signal has much lower information density than language [19], making them insensitive to the order of computing. In the first step, instead of recursive calculation conditioning on previous hidden states (Fig. 2(a)), we compute the hidden state by directly aggregating a set of learnable spatial samplings from the input feature (Fig. 2(b)). In the second step, we conduct spatial alignment through simplifying the matrix generation in SSM equations, which enables a feasible spatial aggregation. The additional benefit is that the simplification facilitates the SSM computation. In the last step, to maintain position awareness, we compute the spatial aggregation among the aggregated pixels and sampling pixels to solve the SSM, thus overcoming the inherent sequential constraints. Specifically, instead of recursive calculation, we generate the hidden state by directly aggregating a set of learnable spatial samplings from the input feature as illustrated in Fig. 2(b).

It is noteworthy that improper sampling operators, such as similarity comparison [49, 68], impose a quadratic complexity, and dense sampling easily leads to training collapse. To maintain efficiency and ease the training burden, we propose a Dynamic State Aggregation (DSA) operator with two variants: the local operator (Lo-DSA) and the spatial-adaptive operator (Sa-DSA). In the shallow layers, the Lo-DSA adopts a pooling layer to aggregate the neighboring image pixels, which aims to enhance local modeling. In the deep layers, the De-DSA learns to predict a set of learnable sampling points from arbitrary spatial locations. We also

propose a sparsification technique, dubbed layer multiplexing, to stabilize the training and reduce the model cost.

We conduct extensive experiments to validate the PV-Mamba on various vision tasks, i.e., image classification [42], object detection [27], and segmentation [70]. In summary, the contributions of this work are threefold:

- We introduce three core steps to parallelize the Mamba pipeline, overcoming the inherent sequential constraints and maintaining high efficiency.
- We develop two variants of DSA operators, which aggregate neighboring and learnable sampling points for the shallow and deep layers. Several practical implementations are proposed, including layer multiplexing.
- Our PVMamba achieves state-of-the-art results in various vision downstream tasks, outperforming recent transformer and Mamba-based vision models.

2. Related Work

State Space Models. State space models (SSMs) [8, 10, 13, 26, 38, 44, 54] are newly emerged model architectures in the deep learning paradigm, with remarkable advances in linear complexity with sequence length. SSMs originated from the continuous state space models [11] in control systems and are applied to the long-sequence modeling problem in language tasks [15, 37]. Later, a series of improvements, such as initialization [9, 44] and normalization techniques [12], further enhance the sequence modeling ability of SSMs. Mamba [8] introduces hardware-optimized parallel scanning (S6), achieving both great efficiency and effectiveness in causal sequence processing. Since then, SSMs have been applied in the vision domain [17, 31, 36, 43, 71] and have shown the potential to be a compelling alternative to CNN/Transformers. ViM [71] and VMamba [31] are pioneers in integrating the Mamba with the modern vision backbone architecture. Before processing by the SSMs, vision Mamba methods have to rearrange the 2D visual components into a 1D sequential format using bidirectional or four-directional scans. The succeeding vision Mamba works [40, 60, 64] optimize the scanning strategies, such as quadtree-based partition [60], to preserve 2D locality and network search [22] to find the optimal scanning. Instead of

improving the scanning strategies, we propose three steps to parallelize the SSM with reasonable assumptions, which overcomes the sequential constraint inherently. Our work provides a vision-specific alternative for the SSMs, which has a hardware-friendly computing pipeline.

Vision Backbone Models. Convolutional Neural Networks (CNNs) [18, 23, 35, 45, 46, 61] and Vision Transformers (ViTs) [5, 33, 48, 50, 52] are two mainstream choices of model architecture, which generally serve as versatile vision backbones for various downstream vision tasks, such as image classification [42], segmentation [70], and object detection [27] and tracking [55]. The CNN was originally developed for image recognition [23], while the transformer was first proposed for language modeling and later adapted into the vision domain for its global modeling ability. The local convolutional operators in CNNs are restricted in global modeling but still prove the necessity of local modeling at the shallow feature layers [59]. In contrast, transformers [49] excel in long-range modeling but at the expense of quadratic computation complexity [33]. Recent Mamba vision models [8, 31, 71] achieve linear complexity with sequence length but show only marginal performance gains compared to transformers. MLLA [16] and VSSD [43] reformulate SSM as a non-causal linear attention model, while our PVMamba still maintains the causal modeling ability. To adapt Mamba for vision, we decouple DSA operators at both shallow and deep levels, creating a hybrid model structure [17].

Dynamic Operators in Vision Models. Dynamic operators [3, 57, 67, 72] have been widely applied in CNN and vision transformer architectures. To enhance the transformation modeling capability of CNNs, the pioneer Deformable Convolution (DCN) series [3, 53, 63, 72] adapts convolution filters to attend to flexible spatial locations by learning a group of offsets. Later, vision transformers [57, 67] adopt dynamic operations to reduce the computation redundancy in the attention scheme. Deformable DETR [73] learns a small set of key sampling points for each query point in the last model stage, which greatly improves the training convergence. For the visual backbone models, PS-ViT [67] and DAT [57] learn a group of spatial sampling points that are shared for the whole feature map in all model stages. Our method can also be viewed as the first vision Mamba model equipped with a spatial adaptive scheme to compute the hidden states for solving the SSM.

3. Method

We first recap the SSM in Sec. 3.1. Then, we elaborate on our two key designs that well-adapt SSM models to visual data, i.e., the parallel state space model (Sec. 3.2) and dynamic state transition (Sec. 3.3). Finally, we depict the overall model architecture in Sec. 3.4.

3.1. Revisiting Selective State Space Model

The core modeling of State Space Models (SSMs) [8, 13] is inspired from the continuous linear time-invariant systems [29]. A hidden state embedding $h(t) \in \mathbb{R}^N$ incorporates the mappings from the input $x(t) \in \mathbb{R}$ to output signal $y(t) \in \mathbb{R}$. The detailed mappings are described by linear ordinary differential equations:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}h(t), \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^{N \times 1}$, and $\mathbf{C} \in \mathbb{R}^{1 \times N}$ are weighting coefficients.

Discretization. For implementation in the deep learning paradigm [39], the continuous system described by Eqn. 1 is discretized using a zero-order hold assumption. It effectively transforms the continuous-time parameters (\mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D}) into corresponding discrete parameters over the specified sampling time-scale $\Delta \in \mathbb{R}^C > 0$:

$$\bar{\mathbf{A}} = e^{\Delta \mathbf{A}}, \bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1} (e^{\Delta \mathbf{A}} - \mathbf{I}) \Delta \mathbf{B}, \bar{\mathbf{C}} = \mathbf{C}. \quad (2)$$

Selective Mechanism. Mamba [8] applies an input-dependent scheme to generate the parameters $\{\bar{\mathbf{B}}_k\}_{k=1}^L$, $\{\bar{\mathbf{C}}_k\}_{k=1}^L$ and $\{\bar{\mathbf{A}}_k\}_{k=1}^L$ from the input sequence $\{x_k\}_{k=1}^L$. Then, it introduces a dynamic selection mechanism to process the sequential states $\{h_k\}_{k=1}^L$. Thus, the output sequence $\{y_k\}_{k=1}^L$ can be calculated with input-dependent parameters:

$$h_k = \bar{\mathbf{A}}_k h_{k-1} + \bar{\mathbf{B}}_k x_k, \quad y_k = \bar{\mathbf{C}}_k h_k. \quad (3)$$

To ease the difficulties of parallelism, Mamba [8] applies hardware-aware optimization when computing the hidden states $\{h_k\}_{k=1}^L$ in Eqn. 3. Mamba2 [4] further unifies the SSM and attention in the concept of state space duality.

Vision SSM. Vision Mamba methods [22, 31, 60, 71] adopt various scanning strategies S , which flatten the 2D image $x \in \mathbb{R}^{H \times W}$ into 1D sequences $x_s \in \mathbb{R}^{L=H \times W}$. To gain global context modeling, vision Mamba generates multiple sequences. Our method dynamically models the spatial context, removing the complex scanning strategies.

3.2. Parallel State Space Model

To overcome the sequential constraints in SSM, we illustrate three core steps in Fig. 3. Our assumption comes from the observation that the information density in visual signal is far more sparse than the language [19], so that the visual modeling is insensitive to the sequential order.

Parallelization. On this basis, we assume that hidden state h_k can be parallelly aggregated by a selection of input pixel embedding x . Fig. 3(a) shows that we replace the previous hidden state h_{k-1} in the SSM equation with the spatial aggregation from input x . Thus, we do not need to compute the hidden states in Eqn. 3 recursively. Then, a spatial

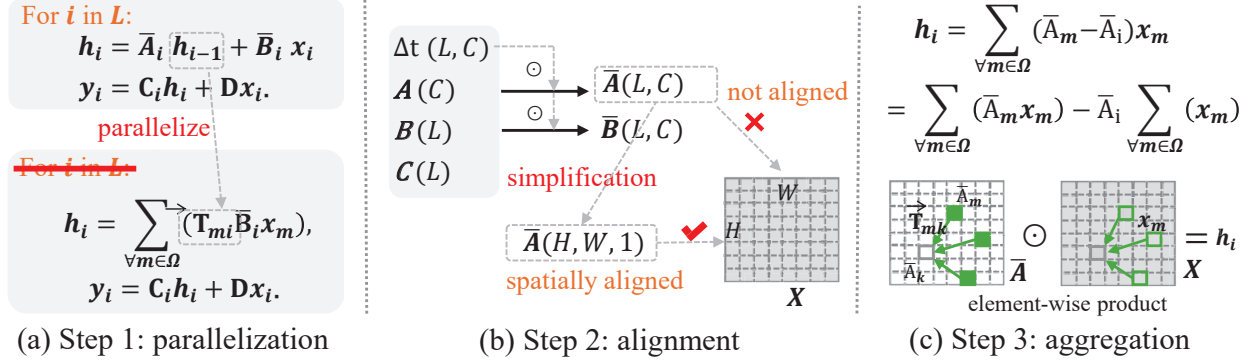


Figure 3. Three core steps of parallelizing the sequential Mamba scheme. (a) shows how we parallelize the sequential SSM solving for hidden states. (b) illustrates the alignment for parallelizing the SSM computation. We simplified parameter generation. For clarity, we omit the batch dimension and other parameter matrices. (c) depicts the computation of transition matrices and spatial aggregation.

sampling operator $\mathcal{P}_\Omega(\cdot)$ is adopted to generate the hidden state h_k instead of recursive computation:

$$h_k = \sum_{i \leq k} \left(\prod_i \bar{A}_i \right) x_k \xrightarrow{\text{Parallelize}} \sum_{\forall w \in \Omega} \mathcal{P}_\Omega(\bar{A}) x_w, \quad (4)$$

where index w belongs to the index set Ω of corresponding embeddings. We omit the \bar{B} for clarity. The details of the index selection operator Ω are in Sec . 3.3.

Alignment. In order to conduct matrix calculation in the parallel SSM, we simplify the matrix \bar{A} in Eqn. 3 from $\mathbb{R}^{L \times C}$ to $\mathbb{R}^{L \times 1}$, where $L = H \times W$. We omit the batch size dimension B here for simplicity. Thus, the matrix parameters in Eqn. 3 can be spatially aligned with the feature $x \in \mathbb{R}^{H \times W}$, which enables a feasible spatial aggregation for the hidden state computation.

Aggregation. In contrast to the cumulative multiplication of transition matrix \bar{A}_k from the previous hidden state h_a to h_b , we reformulate it as a weighted summation which can aggregate the feature states from any two embedding indices:

$$\vec{T}_{ab} = \prod_{i=a}^b \bar{A}_i \xrightarrow{\text{Reformulate}} (\bar{A}_b - \bar{A}_a), \quad (5)$$

where \vec{T}_{ab} is the transition matrix for the transition from the index a to b . A faster approximation, which directly applies \bar{A}_b as the transition matrix, also works well. Then, the current hidden state h_k can be obtained by an aggregation of all the transition states:

$$h_k = \sum_{\forall w \in \Omega} \vec{T}_{wk} x_w. \quad (6)$$

Finally, h_k can be applied to compute the output y_k using the second formula in Eqn. 3. We omit \bar{B} , C for clarity, as it can be calculated by matrix operations.

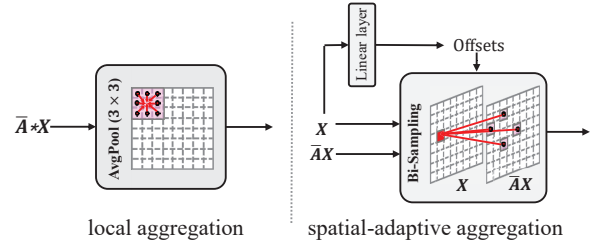


Figure 4. Implementations of local and spatial-adaptive state aggregation.

3.3. Dynamic State Aggregation Layer

With the proposed parallel state space scheme, we describe the DSA layer, which is built upon a vision Mamba block with gated structure [31, 43, 64]. As shown in Fig. 4, we introduce two variants of the DSA operator: local (Lo-DSA) and spatial-adaptive (Sa-DSA) state transition.

Local State Aggregation. In the shallow layers, we focus on enriching the feature representation from the local context in small $K_l \times K_l$ local neighborhoods. The deformable-style $\mathcal{P}_{\Omega_{loc}}$ is transformed to a 2D local operator for each token index x_k and its corresponding 2D coordinates $x_{(i,j)}$. The aggregated state $h_{(i,j)}$ is computed as:

$$h_{(i,j)} = \sum_{\forall (m,n) \in \Omega_{loc}} \vec{T}_{mn} x_{(m,n)}, \quad (7)$$

where Ω_{loc} denotes the indices located at a local window region $\{m \in (i - \frac{K_l-1}{2}, i + \frac{K_l-1}{2}), n \in (j - \frac{K_l-1}{2}, j + \frac{K_l-1}{2})\}$. Neglecting the scalar item, all the hidden states \mathbf{H} can be efficiently implemented by the average pooling layer $\text{Pool}_{(K_l, K_l)}(\cdot)$ with kernel size of $K_l \times K_l$:

$$\mathbf{H} = \text{Pool}_{(K_l, K_l)}(\bar{A} \odot \mathbf{X}), \quad (8)$$

where $\{\bar{A}, \mathbf{H}, \mathbf{X}\}$ are the embeddings of $\{\bar{A}_k, h_k, x_k\}_{k=1}^N$ in 2D format and \odot is the element-wise multiplication operator. We here adopt a faster implementation that directly

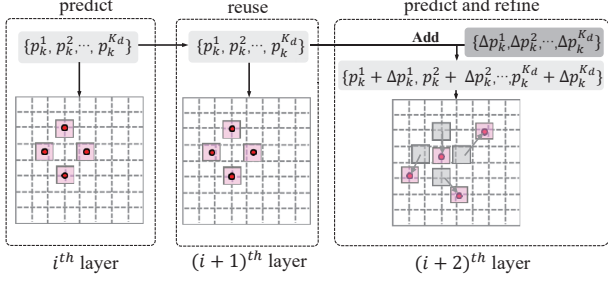


Figure 5. The stacked Sa-DSA layers with layer multiplexing.

applies $\bar{\mathbf{A}}$ as the transition matrix. The implementation details are in the supplementary materials. Our hidden states are efficient parallel forms in contrast to the recurrent forms in Mamba.

Spatial-Adaptive Aggregation. We aim to model the long-range dependencies in the deep layers and enable a dynamic state aggregation among the most relevant regions. To maintain a linear complexity with the length of visual tokens, we adopt a lightweight linear layer $\phi(\cdot)$ to predict sampling points $\{\mathbf{p}_k\}_{k=1}^N \in \mathbb{R}^{2 \times K_d \times N}$ for feature tokens $\{x_k\}_{k=1}^N \in \mathbb{R}^{C \times N}$ instead of a pair-wise similarity comparison:

$$\mathbf{p}_k = \{p_k^1, p_k^2, \dots, p_k^{K_d}\} = \phi(x_k), \quad (9)$$

where K_d is the pre-defined number of sampling points for each visual token x_k . The sampling points \mathbf{p}_k also denote the Ω_{De} for the Sa-DSA layer. Then, to formulate a spatial-adaptive state transition path for each visual token x_k , we adopt a bilinear interpolation and sampling operation **Bi-Samp**(\cdot) to obtain the sampling features and corresponding transition matrices:

$$\{x_k^1, x_k^2, \dots, x_k^{K_c}\} = \mathbf{Bi-Samp}(\mathbf{X}|\mathbf{p}_k), \quad (10)$$

$$\{\bar{\mathbf{T}}_k^1, \bar{\mathbf{T}}_k^2, \dots, \bar{\mathbf{T}}_k^{K_c}\} = \mathbf{Bi-Samp}(\bar{\mathbf{A}}|\mathbf{p}_k) - \bar{\mathbf{A}}, \quad (11)$$

where the **Bi-Samp**(\cdot) operation is differentiable w.r.t. both the embeddings $\{\mathbf{X}, \bar{\mathbf{A}}\}$ and the sampling locations \mathbf{p}_k . In the final, the deformable aggregated hidden state $h_{(i,j)}$ is computed as follows:

$$h_k = \sum_{\forall l \in \Omega_{de}} \bar{\mathbf{T}}_k^l x_k^l. \quad (12)$$

Layer Multiplexing. In practice, we reuse the learned sampling points in stacked Sa-DSA layers, which we call layer multiplexing. As shown in Fig. 5, in a stacked Sa-DSA unit, sampling points \mathbf{p}_k are learned from scratch only in the beginning Sa-DSA layer. Subsequent layers learn to refine or leverage the previous sampling points. The layer multiplexing technique eases the network’s training burden and

reduces the overall computation overhead. Details of the stacked pattern can be found in Sec. 4.4.

Complexity Analysis of DSA Layers. The proposed DSA scheme enables directional sparse modeling while maintaining the linear computation complexity. For Lo-DSA, the implementation using an average pooling layer brings no additional model parameters and acceptable computation overhead. For Sa-DSA, the additional overhead comes from the prediction sub-network for sampling. We remove the independent linear projection, which is widely adopted in deformable operators [53, 72]. Thus, the offsets are directly predicted, resulting in an additional complexity $O(2HWCK_d)$. Typically, consider the tiny model for image classification where $H = W = 14, K_d = 5, C = 512$, the additional computational cost for Sa-DSA in a single block is only 1.003M FLOPs. The layer multiplexing further reduces the overall complexity.

Connections to CNN, Attention, and Graph. Lo-DSA adopts a fixed local operator, which is similar to the depth-wise convolution [66]. Sa-DSA adds learnable offsets to the sampling points, which excels at long-range modeling, like attention [57]. Our DSA scheme is partially inspired by the concept of directed graphs [56, 68]. The difference is that our DSA functions in the hidden states of SSM and dynamically generates the edge weight in a directional style.

3.4. Model Architecture

PVMamba follows a hierarchical architecture from vision transformers [21, 33, 33, 34, 50, 69] and gated block from vision Mamba [31, 43] and Mamba2 [4]. An image $I \in \mathbb{R}^{H_{im} \times W_{im} \times 3}$ is first partitioned into patches, resulting in $N = \lfloor \frac{H_{im}}{4} \rfloor \times \lfloor \frac{W_{im}}{4} \rfloor$ tokens. The tokens are firstly mapped to hidden embeddings with dimension C and then passed into four model stages. In between each stage, a downsampling layer is to reduce the size of feature maps by half and double their channel dimension. Specifically, in the first and second model stages, we stack the Lo-DSA layer. In the third model stage, which consumes the most parameters, we choose the Sa-DSA layers. For layer multiplexing, the first Sa-DSA layer predicts the sampling points at the scratch. Then, two subsequent layers reuse the sampling points, and one last layer refines the points. We build three model variants, i.e., Tiny/Small/Base:

- Tiny: $C = 64$, layer number: {2, 4, 8, 4}
- Small: $C = 64$, layer number: {3, 4, 21, 5}
- Base: $C = 96$, layer number: {3, 4, 21, 5}

4. Experiments

We conduct experiments on various vision benchmarks, i.e., ImageNet-1K [42] for image classification, COCO [27] for object detection, and ADE20K [70] for semantic segmenta-

Cost	Model	Params (M)	FLOPs (G)	Top1-acc (%)
small model ~ 4.5G	CoAtNet-T [35]	29	4.5	82.1
	UniRepLKNet-T [35]	29	4.5	82.1
	ConvNeXt-T [35]	29	4.5	82.1
	MambaoutOut-T	27	4.5	82.7
	InternImage-T [53]	30	5.0	83.5
	DeiT-S [48]	22	4.6	79.9
	Swin-T [32]	29	4.5	81.3
	DAT	29	4.6	82.0
	Vim-S	26	5.1	80.3
	VMamba-T	22	5.6	82.2
	LocalVMamba	26	5.7	82.7
	MLLA-T [16]	25	4.2	83.5
	PVMamba-T	24	4.5	83.9
base model ~ 9.0G	ConvNeXt-S [35]	50	8.7	83.1
	MambaoutOut-S	48	9.0	84.1
	PVTv2-b3 [51]	45	6.9	83.2
	Swin-S [32]	50	8.7	83.0
	DAT-S	50	9.0	83.7
	InternImage-S [53]	50	8.0	84.2
	Swin-S [32]	50	8.7	83.0
	UniFormer-B* [24]	50	8.3	85.1
	VMamba-S	44	11.2	83.5
	PlainMamba-L2	25	8.1	81.6
	LocalVMamba-S	50	11.4	83.7
PVMamba-S	40	7.4	84.2	
large model ~ 18.0G	ConvNeXt-B [35]	89	15.4	83.8
	Swin-B [32]	88	15.4	83.5
	DAT-B	86	17.5	84.0
	PVTv2-L [51]	82	11.8	83.8
	InterImage-B [53]	97	16.0	84.9
	VMamba-B	89	15.4	83.9
	PlainMamba-L3	50	14.4	82.3
	PVMamba-B	89	16.1	84.8

Table 1. Comparison with the state-of-the-art methods on ImageNet-1K classification.

tion. In the following, we compare with the previous state-of-the-art (sota) methods and present ablation studies.

4.1. Image Classification on ImageNet-1K

Settings. We benchmark the proposed PVMamba on ImageNet-1K [42], which contains 1.28M training images and 50K validation images from 1K classes. For a fair comparison, we follow the training settings in [31, 33]. The Top-1 accuracy is reported.

Results. We compare PVMamba against previous CNN, transformer, and Mamba-based SOTA models in Tab. 1. The results in the table show that PVMamba consistently outperforms previous models at different levels. Moreover, PVMamba-S shows a significant improvement over previous Mamba and transformer models, which achieves 84.5% Top1-accuracy with 7.1G FLOPs. It outperforms the VMamba-T [31] and LocalMamba-T [22] by 1.5% and 1.0% in Top1-accuracy, respectively. When scaling up the model size, PVMamba-B also outperforms the VMamba-B [30] by 0.9% in Top1-accuracy.

Backbone	Params FLOPs		Mask R-CNN 1×+MS					
	(M)	(G)	AP^b	AP_{50}^b	AP_{75}^b	AP^m	AP_{50}^m	AP_{75}^m
ResNet-50	44	260	38.2	58.8	41.4	34.7	55.7	37.2
Swin-T	48	267	42.7	65.2	46.8	39.3	62.2	42.2
VMamba-T	50	271	46.5	68.5	52.0	42.1	65.5	45.3
LocalVMamba-T	45	291	46.7	68.7	50.8	42.2	65.7	45.5
PVMamba-T	44	266	47.1	69.3	51.9	42.6	66.5	45.7
ResNet-101	63	336	38.2	58.8	41.4	34.7	55.7	37.2
Swin-S	69	354	44.8	68.6	49.4	40.9	65.3	44.2
VMamba-S	70	349	48.2	69.7	52.5	43.0	66.6	46.4
LocalVMamba-S	69	414	48.4	69.9	52.7	43.2	66.7	46.5
PVMamba-S	60	321	48.6	70.2	53.4	43.7	67.1	47.3
Swin-B	88	496	46.9	-	-	42.3	66.3	46.0
ConvNeXt-B	107	486	47.0	69.4	51.7	42.7	66.3	46.0
VMamba-B	108	485	49.2	71.4	54.0	44.1	68.3	47.7
PVMamba-B	108	509	49.7	71.8	54.4	44.5	68.5	48.1

Table 2. Comparison to other backbones using Mask R-CNN with "1×+MS" schedule.

Backbone	Params FLOPs		Mask R-CNN 3×+MS					
	(M)	(G)	AP^b	AP_{50}^b	AP_{75}^b	AP^m	AP_{50}^m	AP_{75}^m
Swin-T	48	267	46.0	68.1	50.3	41.6	65.1	44.9
ConvNeXt-T	48	262	46.2	67.9	50.8	41.7	65.0	44.9
VMamba-T	50	271	48.8	70.4	53.5	43.7	67.4	47.0
PVMamba-T	44	266	49.1	70.8	53.8	43.9	67.9	47.0
Swin-S	69	354	48.2	69.8	52.8	43.2	67.0	46.1
ConvNeXt-S	70	348	47.9	70.0	52.7	42.9	66.9	46.2
VMamba-S	70	349	49.9	70.9	54.7	44.2	68.2	47.7
LocalVMamba-S	69	414	49.9	70.5	54.4	44.1	67.8	47.4
PVMamba-S	60	321	50.2	80.1	55.0	44.6	68.4	47.9

Table 3. Comparison to other backbones using Mask R-CNN with "3×+MS" schedule.

4.2. Object Detection on COCO

Settings. We conduct object detection and instance segmentation experiments in the COCO 2017 [27] benchmark, which contains 118K, 5K, and 20K images for training, validation, and testing, respectively. We consider two typical object detection settings to validate the effectiveness of our backbone: Mask R-CNN [1] with "1" and "3×+MS" schedule. For these detection frameworks, we utilize similar settings and follow the SwinT [33] and VMamba [31].

Results. We compare PVMamba against SOTA models in Tab. 2 and Tab. 3. The results demonstrate that PVMamba consistently outperforms the CNN/transformer variants and improves the Mamba-based models at different model levels and settings. For the Mask R-CNN with "1×" schedule, PVMamba-T achieves 47.1% AP and 42.6% mask AP, significantly surpassing the VMamba-T [30] by 1.6% AP and 0.5% AP mask with similar model cost. For the Mask R-CNN with "3×" schedule, PVMamba-T/S also outperforms recent sota transformer and Mamba methods, SwinT [33] and VMamba [31].

Backbone	Method	Params(M)	FLOPs(G)	mIoU(%)
Swin-T [33]	UperNet	60M	945G	44.4
InternImage-T [53]	UperNet	59M	944G	47.9
VMamba-T [31]	UperNet	62M	949G	48.0
LocalMamba-T [22]	UperNet	57M	970G	47.9
PVMamba-T	UperNet	54M	941G	48.2
Swin-T [57]	UperNet	81M	1039G	47.6
InternImage-S [53]	UperNet	80M	1017G	50.1
VMamba-S [31]	UperNet	82M	1028G	50.6
LocalMamba-S [22]	UperNet	81M	1095G	50.0
PVMamba-S	UperNet	70M	1004G	50.8
Swin-B [33]	UperNet	121M	1188G	48.1
InternImage-B [53]	UperNet	128M	1185G	50.8
VMamba-B [31]	UperNet	122M	1170G	51.0
PVMamba-B	UperNet	120M	1192G	51.4

Table 4. Comparison with the state-of-the-art on ADE20K.

4.3. Semantic Segmentation on ADE20K

Settings. ADE20K [70] is a widely used semantic segmentation dataset covering 150 semantic categories. It comprises a total of 25K images, with 20K for training, 2K for validation, and 3K for testing. We utilize UperNet [58] in mmseg [2] as our base framework for a fair comparison. More details are in the supplementary materials.

Results. We compare PVMamba with sota CNN/transformer and Mamba models in Tab. 4. The table shows that PVMamba achieves competitive results and outperforms the state-of-the-art models InternImage [53] and Vmamba [31] in pixel-level tasks. PVMamba-T with 48.2% mIoU even surpasses Swin-T [33] with fewer FLOPs.

4.4. Ablation Studies

In this section, we ablate important designs in PVMamba. The ablations adopt image classification, object detection, and semantic segmentation accordingly. To be efficient in ablation studies, we implement a lite version with roughly 13M/2.4G model costs.

DSA Layer. Tab. 5 shows that SSM heavily relies on the additional acceleration [8, 44] to achieve a normal latency (0.613 ms). Our DSA slightly increases the model cost and latency while removing additional complex speeding-up tricks. In Tab. 6, we observe an improvement of 0.5% in Top1-accuracy when replacing the original SS2D with the Sa-DSA operator, showing the effectiveness of the proposed DSA. Moreover, Tab. 6 shows the various improvements from the existing well-designed modules, such as the embedding and MESA [6].

Sampling Points. The core hyperparameter is the number of sampling points for each pixel, which analyzes the impacts on both classification and dense prediction tasks in Tab. 7. For image classification in ImageNet [42], the Top1-accuracy in ImageNet [42] begins to saturate around 82.2% when the number of points is larger than 4. For dense prediction tasks in COCO [27], the improvement increases

Model	Pipeline	Acceleration	Params (M)	FLOPs (G)	Latency (ms)
SSM	sequential	w/o	32	5.3	212×10^3
SSM	sequential	w associative scan [44] w hardware optim. [8]	32	5.3	0.613
DSA	parallel	w/o	34	5.5	0.702

Table 5. Efficiency comparison between SSM and DSA.

Step	Method	Params (M)	FLOPs (G)	Top1 (%)
0	Baseline	30	4.8	82.5
1	SS2D \rightarrow Sa-DSA	31	4.9	83.0
2	Hybrid model stage	30	4.8	83.2
3	{Block, Channel Number} \downarrow	23	4.2	83.4
4	Simple Embed. \rightarrow More Embed. layers	23	4.2	83.5
5	+ MESA	24	4.5	83.9

Table 6. The ablation experiments demonstrate the roadmap from the Mamba-based vision backbone to PVMamba. The tiny model is adopted.

K_d	Params (M)	FLOPs (G)	Memory Cost (512 img)	Top1 (%)	AP_b (%)	AP_m (%)
3	13.5	2.32	5.8GB	82.1	45.2	40.9
4	13.5	2.32	5.8GB	82.2	45.4	41.0
5	13.5	2.33	5.8GB	82.2	45.4	41.2
7	13.5	2.34	5.8GB	82.2	45.6	41.3
9	13.5	2.35	6.5GB	82.3	45.7	41.5

Table 7. Comparison of performance and speed across various sampling point settings in the lite model. The size of the testing image is 224×224 .

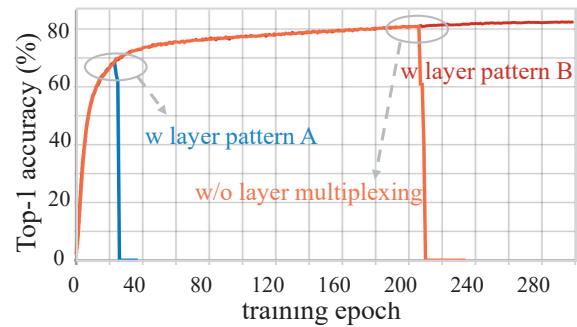


Figure 6. Training curves of w pattern A (collapse), w pattern B (stable), and w/o layer multiplexing (collapse) technique.

more consistently when the number of points rises. For the memory cost and model efficiency, the memory cost only rises significantly to 6.5 GB when the number of points ($K_d = 9$) is too large, while the model efficiency decreases more linearly. Considering the balance of performance in various downstream tasks and efficiency, the $K_d = 5$ can be set for performance, and $K_d = 3$ for efficiency.

Layer Multiplexing. The layer multiplexing technique is

Stage1	Stage2	Stage3	Stage4	Params (M)↓	FLOPs (G)↓	Throughput (img/s)↑	Top1 (%)
Sa-DSA	Sa-DSA	Sa-DSA	Sa-DSA	14.9	2.42	1128	82.0
Lo-DSA	Lo-DSA	Sa-DSA	Sa-DSA	14.9	2.39	1423	82.2
Lo-DSA	Lo-DSA	Sa-DSA	Attention	13.5	2.32	1524	82.2

Table 8. The performance and efficiency comparisons on different hybrid model stage configurations.

Model	Params (M)	FLOPs (G)	Throughput (imgs/s)	Top1 (%)
ConvNeXt-T [35]	29	4.5	775	82.9
Focal-T [65]	29	4.9	582	82.2
Swin-S [33]	50	8.8	1006	83.0
PVMamba-Tiny	24	4.5	1038	83.9
ConvNeXt-S [35]	50	8.7	447	83.1
Focal-S [65]	51	9.1	351	83.5
PVMamba-Small	40	7.4	611	84.2
ConvNeXt-B [35]	89	15.4	293	83.8
Focal-B [65]	90	16.4	256	84.0
PVMamba-Base	89	16.1	405	84.8

Table 9. Comparison of inference speed. We measure with an A800 GPU platform.

key to applying our DSA scheme to the visual backbone. As shown in Fig. 6, a training collapse phenomenon is observed when applying DSA operators without layer multiplexing. The potential reason is that the dense sampling is easy to overfit, as the semantic-meaning area is limited. It indicates that the Sa-DSA operator requires a sparsification technique to ease the learning burden. Multiplexing the sampling points in previous layers can significantly ease the risk of overfitting, which also reduces the overall computational cost and improves convergence to a degree. We also explore the pattern of multiplexing: pattern A refers to learning the sampling points in the beginning and multiplexing them in the remaining layers; pattern B is an interleaved style that stacks several repeated units. Thus, each unit can learn the sampling points separately. The training curve of pattern B successfully overcomes the collapse, validating the effectiveness of the interleaved design.

Hybrid Operator. The effectiveness of the cooperation of different operators in four model stages is validated in Tab. 8. Restricting the dynamic operator into the neighboring window in the first two model stages improves the 0.2% Top1-accuracy in ImageNet [42] classification. In the fourth model stage, which generally includes 2 layers and the smallest spatial size of the feature map, we replace the DSA with vanilla attention [5, 43], which has negligible negative impacts on the performance and model cost while raising the throughput from 1423 to 1524 img/s.

Inference Speed. We compare the PVMamba’s inference speed with the recent strong CNN/transformer backbones in Tab. 9. Our models show competitive inference speed and superior performance across various model levels.

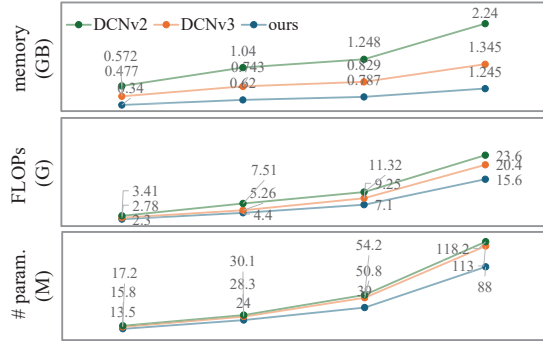


Figure 7. Model cost and GPU memory cost (32 images with the size of 224×224) when scaling up.

Scaling Capability. We compare the scaling capability of different deformable operator implementations: DCNv2-style [72], DCNv3-style [53], and ours. In Fig. 7, our implementation shows superior scaling capability with high model efficiency and performance. Our model consumes nearly less 52.6% GPU memory than the DCNv2-style in the base model. It is attributed to the lightweight design in PVMamba, in which we remove the independent projection layers for weight and offset in DCNv2 [72] and further omit the in-out projection layers in DCNv3 [53].

5. Limitations

The number of sampling points can be increased to enhance performance at the cost of lower speed. Our PVMamba shares similarity with the DCN series [3, 53, 63], which could benefit from hardware optimizations in future work. Though PVMamba exhibits strong performance and competitive efficiency compared to the previous methods, the application to the multi-modality task, i.e., text [8], image [31], and video data [25], remains under-explored.

6. Conclusion

This paper parallelizes the vision Mamba with a dynamic state aggregation scheme, which can naturally handle the 2D visual data. PVMamba is the first parallel SSM variant equipped with a spatial adaptive operator, overcoming the inherent sequential constraints. PVMamba achieves state-of-the-art performance on various vision downstream tasks, such as classification, detection and semantic segmentation, significantly surpassing previous transformer and Mamba-based methods. We hope that PVMamba will provide an insightful solution to bridge the gap between sequence-based language models and vision models.

Acknowledgments. This work was supported by NSFC (62322113, 62376156), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and the Fundamental Research Funds for the Central Universities.

References

- [1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 6
- [2] MMSegmentation Contributors. Mmsegmentation, an open source semantic segmentation toolbox, 2020. 7
- [3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 3, 8
- [4] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *ICML*, 2024. 3, 5
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1, 3, 8
- [6] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In *NeurIPS*, 2022. 7
- [7] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, Youqiang Zhang, and Junshi Huang. Dimba: Transformer-mamba diffusion models. *arXiv:2406.01159*, 2024. 1
- [8] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *ICML*, 2024. 1, 2, 3, 7, 8
- [9] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In *NeurIPS*, 2020. 1, 2
- [10] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv:2111.00396*, 2021. 2
- [11] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *NeurIPS*, 2021. 2
- [12] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *NeurIPS*, 2022. 2
- [13] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022. 1, 2, 3
- [14] Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. Mambair: A simple baseline for image restoration with state-space model. In *ECCV*, 2025. 1
- [15] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Neurips*, 2022. 2
- [16] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. In *NeurIPS*, 2025. 3, 6
- [17] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv:2407.08083*, 2024. 2, 3
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3
- [19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 2, 3
- [20] Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes Fischer, and Bjorn Ommer. Zigma: Zigzag mamba diffusion model. In *ECCV*, 2024. 1
- [21] Tao Huang, Lang Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Lightvit: Towards light-weight convolution-free vision transformers. *arXiv:2207.05557*, 2022. 5
- [22] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv:2403.09338*, 2024. 1, 2, 3, 6, 7
- [23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 3
- [24] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. In *ICLR*, 2022. 6
- [25] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *ECCV*, 2025. 1, 8
- [26] Shufan Li, Harkanwar Singh, and Aditya Grover. Mamband: Selective state space modeling for multi-dimensional data. In *ECCV*, 2024. 1, 2
- [27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 3, 5, 6, 7
- [28] Jiarun Liu, Hao Yang, Hong-Yu Zhou, Yan Xi, Lequan Yu, Cheng Li, Yong Liang, Guangming Shi, Yizhou Yu, Shaoting Zhang, et al. Swin-umamba: Mamba-based unet with -based pretraining. In *MICCAI*, 2024. 1
- [29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 3
- [30] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *Neurips*, 2024. 6
- [31] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. In *NeurIPS*, 2024. 1, 2, 3, 4, 5, 6, 7, 8
- [32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 6
- [33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 3, 5, 6, 7, 8

- [34] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 5
- [35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, et al. A convnet for the 2020s. In *CVPR*, 2022. 3, 6, 8
- [36] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv:2401.04722*, 2024. 1, 2
- [37] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *ICLR*, 2023. 2
- [38] Eric Nguyen, Karan Goel, Albert Gu, Gordon W Downs, Preey Shah, Tri Dao, Stephen A Baccus, and Christopher Ré. S4nd: Modeling images and videos as multidimensional signals using state spaces. In *NeurIPS*, 2022. 1, 2
- [39] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Workshop*, 2017. 2, 3
- [40] Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. In *NeurIPS*, 2024. 2
- [41] Jiacheng Ruan and Suncheng Xiang. Vm-unet: Vision mamba unet for medical image segmentation. *arXiv:2402.02491*, 2024. 1
- [42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large scale visual recognition challenge. *IJCV*, 115:211–252, 2015. 2, 3, 5, 6, 7, 8
- [43] Yuheng Shi, Mingjing Dong, Mingjia Li, and Chang Xu. Vssd: Vision mamba with non-casual state space duality. *arXiv:2407.18559*, 2024. 2, 3, 4, 5, 8
- [44] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint*, 2022. 2, 7
- [45] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3
- [46] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 3
- [47] Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv:2405.14224*, 2024. 1
- [48] Hugo Touvron, Matthieu Cord, Matthijs Douze, et al. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 3, 6
- [49] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 3
- [50] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 3, 5
- [51] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10, 2022. 6
- [52] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. In *ICLR*, 2022. 3
- [53] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *CVPR*, 2023. 3, 5, 6, 7, 8
- [54] Robert L Williams, Douglas A Lawrence, et al. *Linear state-space control systems*. John Wiley & Sons, 2007. 1, 2
- [55] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013. 3
- [56] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *TNNLS*, 32(1):4–24, 2020. 5
- [57] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, 2022. 3, 5, 7
- [58] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 7
- [59] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. In *NeurIPS*, 2021. 3
- [60] Fei Xie, Weijia Zhang, Zhongdao Wang, and Chao Ma. Quadmambata: Learning quadtree-based selective scan for visual state space model. In *NeurIPS*, 2024. 2, 3
- [61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 3
- [62] Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. In *MICCAI*, 2024. 1
- [63] Yuwen Xiong, Zhiqi Li, Yuntao Chen, Feng Wang, Xizhou Zhu, Jiapeng Luo, Wenhai Wang, Tong Lu, Hongsheng Li, Yu Qiao, et al. Efficient deformable convnets: Rethinking dynamic and sparse operator for vision applications. In *CVPR*, 2024. 3, 8
- [64] Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. In *arXiv:2403.17695*, 2024. 2, 4
- [65] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021. 1, 8
- [66] Weihao Yu, Pan Zhou, Shuicheng Yan, and Xinchao Wang. Inceptionnext: when inception meets convnext. In *CVPR*, 2023. 5

- [67] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *CVPR*, 2021. 3
- [68] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *NeurIPS*, 2019. 2, 5
- [69] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *CVPR*, 2021. 5
- [70] Bolei Zhou, Hang Zhao, Xavier Puig, et al. Scene parsing through ade20k dataset. In *CVPR*, 2017. 2, 3, 5, 7
- [71] Lianghai Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. In *ICML*, 2024. 1, 2, 3
- [72] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 3, 5, 8
- [73] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 3