arXiv:2503.21262v3 [cs.CV] 30 Mar 2026

vGamba: Attentive State Space Bottleneck for efficient Long-range Dependencies in Visual Recognition

Yunusa Haruna[1], Adamu Lawan[1,2,3], Shamsuddeen Hassan Muhammad[4], Jiaquan Zhang[2,3], and Chaoning Zhang[3]

[1] NewraLab, Suzhou, China
[2] Beihang University, Beijing, China
[3] Beijing GoerTek Alpha Labs, Beijing, China
[4] Imperial College London, UK
[5] University of Electronic Science and Technology of China

Abstract. Capturing long-range dependencies (LRD) efficiently is a core challenge in visual recognition, and state-space models (SSMs) have recently emerged as a promising alternative to self-attention for addressing it. However, adapting SSMs into CNN-based bottlenecks remains challenging, as existing approaches require complex pre-processing and multiple SSM replicas per block, limiting their practicality. We propose vGamba, a hybrid vision backbone that replaces the standard bottleneck convolution with a single lightweight SSM block, the Gamba cell, which incorporates 2D positional awareness and an attentive spatial context (ASC) module for efficient LRD modeling. Results on diverse downstream vision tasks demonstrate competitive accuracy against SSM-based models such as VMamba and ViM, while achieving significantly improved computation and memory efficiency over Bottleneck Transformer (BotNet). For example, at 2048×2048 resolution, vGamba is $2.07\times$ faster than BotNet and reduces peak GPU memory by 93.8% (1.03 GB vs. 16.78 GB), scaling near-linearly with resolution comparable to ResNet-50. These results demonstrate that Gamba Bottleneck effectively overcomes the memory and compute constraints of BotNet global modeling, establishing it as a practical and scalable backbone for high-resolution vision tasks.

Keywords: Attention Mechanism · Bottleneck · CNN · SSM

1 Introduction

Modeling long-range dependencies (LRD) is essential in visual recognition for interpreting complex scenes where objects span large spatial regions, such as in instance segmentation or aerial imagery. Traditional convolutional neural network (CNN) backbones have dominated the field [3,15,27,30,31], and while their hierarchical structure allows effective feature extraction, their strong inductive bias and local receptive fields (RF) limit their ability to model LRD. To mitigate this, larger kernels and dilated convolutions have been used to expand the

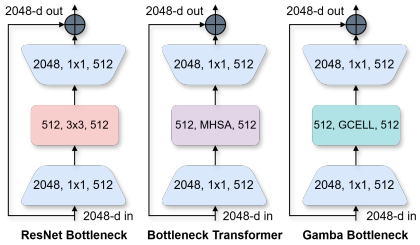


Fig. 1: Bottleneck evolution: CNN, Transformer and SSM. BoTNet replaces the 3×3 convolution with MHSA while vGamba replaces it with GCell. At 2048×2048 resolution, vGamba is $2.07\times$ faster than BoTNet and reduces peak GPU memory by 93.8% (Sec. 5.4).

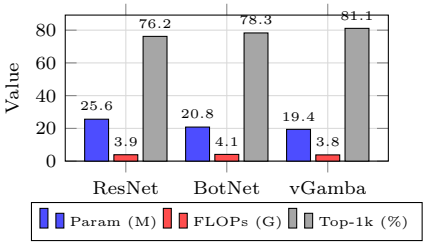


Fig. 2: Quantitative comparison of bottleneck designs. vGamba gains +2.8% Top-1 over BoTNet with 1.4M fewer parameters and 0.3G lower FLOPs, showing that the proposed GCell achieves better representational capacity at lower computational cost (Sec. 5.1).

RF; however, larger kernels are computationally expensive, and dilation causes spatial discontinuities that reduce smooth LRD modeling [7, 39]. The evolution of bottleneck designs from CNNs to Transformers and more recently to State-Space Models (SSMs) is illustrated in Fig. 1. While BoTNet improves LRD, its computational overhead is higher. A quantitative comparison of representative bottleneck designs in terms of parameters, FLOPs and accuracy is provided in Fig. 2, highlighting the trade-offs between efficiency and performance.

Consequently, Vision Transformers (ViT) [8], [12] have gained popularity for their ability to capture LRD using self-attention (SA) mechanisms, which allow them to model global context more effectively than CNN, as shown in the spatial transport and convergence results in Fig. 3. However, the computational cost of SA scales quadratically with sequence length, making ViT computationally expensive, especially for high-resolution images and real-time applications. Various techniques, such as low-rank factorization [38], linear approximation [23,28], and sparse attention [2], have been proposed to reduce computational costs while preserving performance. However, these approaches often involve trade-offs between efficiency and effectiveness.

State-space models (SSM) [11] have recently emerged as an alternative to SA mechanisms, offering a more efficient way to model LRD. By capturing sequential dynamics in a compact form, SSMs reduce computational overhead while maintaining competitive performance. Mamba [10], a selective SSM, was introduced to further optimize efficiency, achieving up to $5\times$ higher throughput on sequence modeling benchmarks by selectively retaining relevant information and reducing redundant computations. However, adapting SSMs for vision tasks remains challenging, as it requires specific spatial pre-processing. For instance, VMamba [19] employs cross-scan and merging strategies requiring four SSM replicas per block, while ViM [41] introduces bidirectional SSMs requiring two replicas per forward pass. Although effective, these designs are not suited as drop-in replacements for the single 3×3 convolution in a CNN bottleneck, as the added replicas increase architectural complexity and memory overhead. This motivates the need for a

lightweight, single-block SSM design that integrates directly into standard CNN pipelines. See Fig. 4 for architectural illustrations.

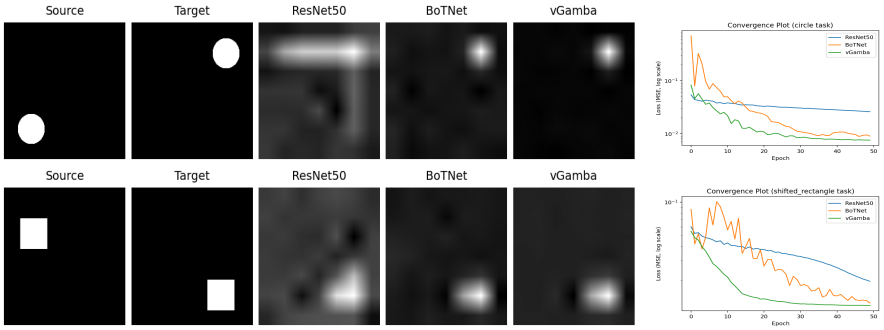


Fig. 3: To test the models' ability to capture LRD, we transport information between distant spatial diagonal locations, placing the source and target in opposite corners using different shapes. We compare three bottleneck architectures ResNet-50 (CNN), BotNet (Transformer), and vGamba (SSM) all trained under identical settings (AdamW, lr 0.001, 50 epochs, 224^2, MSE). ResNet-50 shows an almost straight-line convergence curve, indicating limited capacity to model LRD. BotNet achieves better LRD capture than ResNet-50, yet at a higher computational cost due to its quadratic SA complexity. In contrast, vGamba converges faster and more smoothly, producing more accurate reconstructions than both ResNet-50 and BotNet, effectively capturing LRD and spatial transport patterns at lower computational cost.

To this end, we introduce vGamba, inspired by the Transformer bottleneck [29], we design a lightweight bottleneck module that integrates into the final layer of ResNet to efficiently capture LRD through a single SSM block. vGamba is a hybrid vision backbone built upon the Gamba Block, a compact structure derived from CNN bottlenecks. The Gamba cell extends Mamba for vision tasks by adapting it to 2D spatial structures and integrating ASC for enhanced spatial awareness. By combining these components, vGamba improves representational efficiency and contextual understanding, serving as an efficient alternative to conventional vision backbones. The contributions of this work are stated below:

- We identify that existing SSM-based vision models require complex preprocessing and multi-SSM stacking, making them unsuitable as drop-in replacements for the 3×3 convolution in CNN bottlenecks. We address this with the Gamba bottleneck, a lightweight single-block SSM design with 2D positional awareness and an ASC module for efficient LRD modeling.
- vGamba achieves competitive accuracy against SSM-based models including VMamba and ViM across classification, detection, and segmentation, while using fewer parameters and lower FLOPs (Sec. 5).

– vGamba demonstrates near-CNN-level scaling efficiency: at 2048×2048 resolution it is $2.07\times$ faster than BoTNet and reduces peak GPU memory by 93.8%, making it practical for high-resolution vision tasks (Sec. 5.4).

2 Related Work

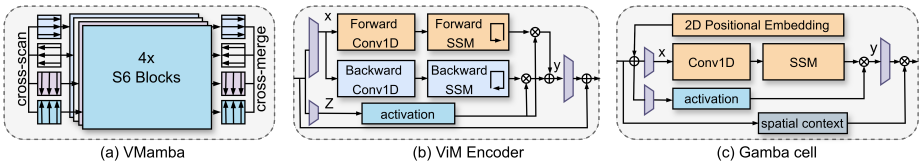


Fig. 4: Schematic comparison of the proposed Gamba cell (c) against (a) VMamba [19] and (b) ViM [41] as drop-in replacements for the 3×3 convolution in the final ResNet bottleneck stage. VMamba requires four SSM replicas per block and ViM requires two, incurring higher complexity and FLOPs. Under identical settings, vGamba-B outperforms VMamba-B by +1.8% and ViM-B by +0.6% in Top-1 accuracy, while using 6.11M and 2.60M fewer parameters respectively, demonstrating the best efficiency-accuracy trade-off with a single SSM block (Appendix A.3.).

Efforts to capture LRD in visual data have evolved across several paradigms. Early CNN-based approaches focused on enlarging the RF by using larger kernels or dilated convolutions. For instance, the Global Convolutional Network [26] used large kernels to model LRD, while the Dilated Residual Network [39] leveraged dilation to expand the RF without increasing model depth or complexity. While both approaches are effective, the former increases computational cost, whereas the latter can introduce gridding artifacts. An alternative is the Non-Local Neural Network [33], which mitigates this by computing responses as weighted sums over all positions, effectively capturing spatial–temporal LRD, but at high computational cost.

Building on this direction of global modeling, ViT [8] modeled global context from the first layer, treating image patches as tokens and linking them through SA, effectively capturing LRD. However, their quadratic complexity and large data demands make them resource-intensive. Several works addressed this: EfficientFormer [38] reduced backpropagation cost via low-rank projections, while LUNA [23] and UFO-ViT [28] proposed linear variants. SparseViT [2] further improved efficiency by pruning inactive tokens. Despite these advances, SA-based models remain costly for high-resolution visual tasks.

More recently, Mamba [10] applied SSMs to sequence modeling, offering efficiency and scalability beyond traditional attention. Originally designed for autoregressive tasks, its 1D structure posed challenges for 2D vision. VMamba [19] extended Mamba with cross-scan and cross-merge operations to capture pixels

in all directions, creating four replicas of the S6 blocks. ViM [41] introduced bidirectional SSMs, creating forward and backward blocks to better capture LRD in images, enabling effective visual representation learning. These adaptations demonstrate that SSMs can provide an efficient alternative to ViTs, but at the cost of architectural complexity. However, integrating SSMs into standard CNN pipelines remains challenging. Existing SSM-based models require heavy preprocessing such as cross-scan, cross-merge, or bidirectional replication, resulting in multiple SSM instances per block and increased architectural complexity, Fig. 4.

To achieve a balance between efficiency and simplicity, it becomes essential to explore lightweight mechanisms that can embed sequence modeling within CNN backbones without architectural overhead. Inspired by Bottleneck Transformers [29], which successfully incorporate MHSA into CNN, we aim to bring SSM into CNN pipelines in a similarly efficient and modular manner.

3 Preliminaries

SSM represents an input sequence $x(t) \in \mathbb{R}$ through a latent state $h(t) \in \mathbb{R}^N$ using the following continuous linear system:

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t), \\ y(t) &= Ch(t) + Dx(t), \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{N\times N}$ is the state transition matrix, $B \in \mathbb{R}^{N\times 1}$ maps the input to the state, $C \in \mathbb{R}^{1\times N}$ maps the state to the output, and $D \in \mathbb{R}$ is a feedthrough term.

To apply SSMs in deep learning, the continuous system is discretized using the Zero-Order Hold (ZOH) method, which assumes the input remains constant over each time step Δt. The discretized system is defined as:

$$\hat{A} = e^{A\Delta t}, \quad \hat{B} = \left(\int_0^{\Delta t} e^{A\tau} d\tau \right) B, \tag{2}$$

leading to the discrete recurrence:

$$\begin{aligned} h_k &= \hat{A}h_{k-1} + \hat{B}x_k, \\ y_k &= Ch_k + Dx_k. \end{aligned} \tag{3}$$

Earlier deep SSM variants such as S4 [11] introduced efficient parameterizations and fast discretizations for long-range modeling. Mamba [10] extends these formulations by making the parameters B, C, and the discretizations scale input-dependent. This dynamic adaptation allows the model to selectively retain relevant information and efficiently model LRD, achieving strong scalability and efficiency beyond traditional attention mechanisms. However, Mamba's causal formulation is designed for sequential data, making direct application to 2D visual inputs non-trivial, a limitation addressed by the proposed Gamba cell (Sec. 4.1).

4 Method

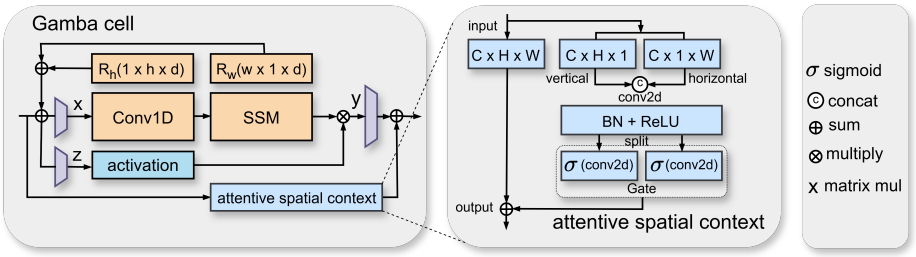


Fig. 5: Gamba Bottleneck showing Gamba Cell and the ASC module.

Problem Statement. Existing SSM adaptations for visual tasks, such as VMamba [19] and ViM [41], rely on cross-scan, cross-merge, or bidirectional operations to adapt 1D SSMs to 2D spatial structures, enabling LRD modeling. While effective at modeling global context, these designs require multiple SSM replicas per block, significantly increasing memory usage and architectural complexity, see Fig. 4. This makes them unsuitable as drop-in replacements for the standard 3×3 convolution in CNN bottlenecks, which are designed for efficiency, modularity, and simplicity.

Goal. Our objective is to adapt Mamba-style SSMs to fit directly into a CNN bottleneck, replacing the 3×3 convolution with a single forward SSM pass per block. This design aims to efficiently capture LRD within a standard CNN pipeline while maintaining low computational and memory overhead. By integrating SSMs in a modular and lightweight manner, we aim to surpass the effectiveness of traditional CNN and reduce the memory and computational demands of Bottleneck Transformers [29], whose MHSA has quadratic complexity.

4.1 Gamba Cell

The original Mamba model relies on causal state transitions, where the i-th token depends only on tokens up to position $(i-1)$ (Eq. 3). While this is well-suited for auto-regressive tasks, it introduces long-range decay in visual processing, weakening spatial dependencies over distance. Unlike VMamba [19] and ViM [41], which mitigate this by replicating the SSM block in multiple directions, the Gamba cell preserves the single causal SSM pass and compensates through 2D Relative Positional Embeddings (RPE) [35] and the ASC module, maintaining efficiency without sacrificing spatial context. The Gamba cell is shown in Fig. 5.

Spatial Sequence Encoding. Given an input feature $X \in \mathbb{R}^{B\times C\times H\times W}$, the Gamba cell proceeds as follows: (i) reshape X to a sequence $X_{\text{seq}} \in \mathbb{R}^{B\times (HW)\times C}$ and inject 2D RPE, (ii) pass through the Mamba block M, (iii) apply the ASC module for spatial context refinement (Fig. 6), and (iv) reshape back to $X_{\text{out}} \in \mathbb{R}^{B\times C\times H\times W}$.

Specifically, spatial priors are injected via RPE $P = R_h + R_w \in \mathbb{R}^{1\times C\times (HW)}$, where R_h and R_w encode positional information along the height and width axes respectively. The position-enriched sequence is:

$$X_{\text{seq}} = X_{\text{seq}} + P^T. \tag{4}$$

This compensates for the loss of 2D spatial structure during flattening, providing the SSM with explicit positional context prior to the causal scan.

SSM Processing. The enriched sequence is passed through the Mamba block M, resulting:

$$X_{\text{mamba}} = M(X_{\text{seq}}), \tag{5}$$

which captures LRD across the spatial sequence. While the causal scan limits direct bidirectional interaction, the preceding RPE and the subsequent ASC module collectively recover fine-grained spatial context, as detailed in Sec. 4.2. The output is reshaped back to $X_{\text{out}} \in \mathbb{R}^{B\times C\times H\times W}$.The full algorithm of the Gamba cell is provided in Appendix A.4.

4.2 ASC Module

While SSM captures LRD through global interactions, adding 2D-RPE alone is insufficient for precise 2D spatial modeling. To address this, we introduce ASC, a lightweight attentive module that enhances fine-grained spatial context while preserving global interactions, without the need to reapply the SSM block. Fig. 6 illustrates ASC implementation. Unlike the original Coordinate Atten-

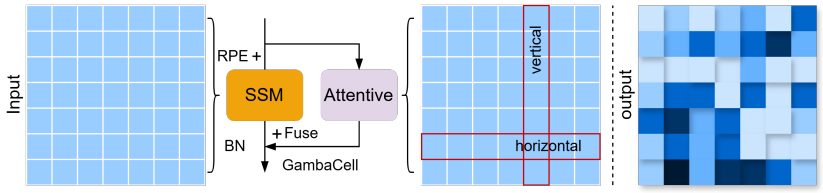


Fig. 6: ASC fuses SSM and RPE to form the Gamba cell.

tion [16], which encodes spatial coordinates by separately aggregating horizontal and vertical features and fusing them uniformly, we extend this design to capture directional asymmetry and channel-dependent context sensitivity. Specifically, ASC introduces per-channel adaptive gating and axis-wise learnable biases, enabling each feature channel to dynamically balance horizontal and vertical information while encoding asymmetric spatial priors. Given an input feature

$X \in \mathbb{R}^{B\times C\times H\times W}$, ASC performs coordinate pooling along height (x_h) and width (x_w), followed by shared transformations and gated fusion:

$$f_h = \mathrm{Conv}_h(x_h) + b_h, \tag{6}$$

$$f_w = \mathrm{Conv}_w(x_w) + b_w, \tag{7}$$

$$\mathrm{ASC}(X) = X \odot \sigma\big(\alpha f_h + (1-\alpha) f_w\big), \tag{8}$$

where $\alpha \in \mathbb{R}^{1\times C\times 1\times 1}$ is the learnable per-channel gate, b_h and b_w are axis-wise learnable biases, and $\odot$ denotes element-wise multiplication.

These extensions allow ASC to capture fine-grained and asymmetric spatial dependencies along both axes efficiently, improving localized context modeling without the overhead of repeated SSM blocks.

Memory and IO Efficiency To capture LRD, BoTNet [29] replaces the 3×3 convolution in ResNet-50 bottlenecks with MHSA. However, due to the quadratic memory complexity $\mathcal{O}(M^2)$ of MHSA with respect to sequence length M, it is restricted to the final stage (7×7 resolution) to avoid memory explosion.

In contrast, our Gamba bottleneck integrates a Mamba-style selective SSM, whose computational complexity scales linearly with sequence length $\mathcal{O}(M)$. Moreover, we adopt the IO-aware selective scan implementation of Mamba, which reduces memory IO from $\mathcal{O}(BMEN)$ to $\mathcal{O}(BME + EN)$ by performing state updates entirely in SRAM and writing back only the output activations. Since the Gamba cell is introduced only in the fourth low-resolution stage, where $M = \frac{HW}{32^2}$, global modeling is achieved with controlled memory and computational overhead, preserving the efficiency of the original CNN backbone, with empirical validation of this scaling behavior provided in Sec. 5.4.

Computation Efficiency The Gamba block leverages SSM to capture LRD efficiently, similar to SA in Transformers. Given a visual sequence $T \in \mathbb{R}^{1\times M\times D}$ with default $E = 2D$, the computational complexities are:

$$\Omega(\text{self-attention}) = 4MD^2 + 2M^2D, \tag{9}$$

$$\Omega(\text{SSM}) = 3M(2D)N + M(2D)N, \tag{10}$$

$$\Omega(\text{CNN}) = M \cdot K^2 \cdot D^2. \tag{11}$$

Here, SA scales quadratically with sequence length M, whereas SSM scales linearly, enabling efficient processing of long sequences. Unlike CNN that capture local context efficiently, but capturing global interactions requires deeper stacks or larger RF, increasing memory and computation compared to SSM-based approaches like vGamba.

4.3 Architectural details

The first three stages capture local details and perform low-level feature extraction, where the input $x \in \mathbb{R}^{C\times H\times W}$ is processed through a stem layer and

standard ResNet bottleneck blocks [15]. These layers extract hierarchical features while progressively reducing spatial dimensions:

$$x_1 \in \mathbb{R}^{C_1 \times \frac{H}{4} \times \frac{W}{4}}, \quad x_2 \in \mathbb{R}^{C_2 \times \frac{H}{8} \times \frac{W}{8}}, \quad x_3 \in \mathbb{R}^{C_3 \times \frac{H}{16} \times \frac{W}{16}}$$

In stage 4, we replace the 3×3 convolutional block with Gamba cell, enhancing LRD and global context modeling. The hybrid vGamba captures feature representations at deeper levels:

$$x_4 \in \mathbb{R}^{C_4 \times \frac{H}{32} \times \frac{W}{32}}$$

Finally, global average pooling (GAP) and a fully connected layer (FC) produce the final classification output:

$$y = \mathrm{FC}(\mathrm{GAP}(x_4))$$

More architectural details of vGamba are provided in Appendix A.4.

5 Experiments

We evaluated vGamba on ImageNet-1K for classification (Section 5.1), ADE20K for segmentation (Section 5.2), and COCO for detection (Section 5.3). Then qualitative experiments on the AID dataset (Section 5.5) and performed ablation studies (Section 5.6). Experiments are conducted on NVIDIA GeForce RTX 4090 GPU, featuring 24 GB of GDDR6X VRAM, CUDA Compute Capability 8.9, and driver version 570.144 with CUDA 12.8 support.

Table 1: Comparison on ImageNet-1K.

IMAGENET-1K				
Method	Size	FLOPs (G)	Param (M)	Top-1 (%)
ResNet-50 [15]	224^2	3.9	25	76.2
ResNet-101 [15]	224^2	7.6	45	77.4
ResNet-152 [15]	224^2	11.3	60	78.3
BoT50 [29]	224^2	4.1	20.8	78.3
EffNet-B2 [31]	224^2	1.0	9	80.1
EffNet-B3 [31]	300^2	1.8	12	81.6
RegNetY-4GF [27]	224^2	4.0	21	80.0
RegNetY-8GF [27]	224^2	8.0	39	81.7
ViT-B/16 [8]	384^2	55.4	86	77.9
ViT-L/16 [8]	384^2	190.7	307	76.5
DeiT-S [32]	224^2	4.6	22	79.8
DeiT-B [32]	224^2	17.5	86	81.8
Swin-T [20]	224^2	4.5	29	81.3
CoaT-Lite-S [4]	224^2	4.1	19.8	82.3
CrossViT-B [1]	240^2	20.1	105	82.2
Vim-S [41]	224^2	5.3	26	80.3
Vim-B [41]	224^2	–	98	81.9
EffVMamba-T [25]	224^2	0.8	6	76.5
EffVMamba-S [25]	224^2	1.3	11	78.7
S4ND-ViT-B [24]	224^2	17.1	89	80.4
VMamba-T [19]	224^2	4.9	30	82.6
MambaVision-T [13]	224^2	4.4	31.8	82.3
vGamba-B	224^2	3.8	19.4	81.1
vGamba-L	224^2	7.6	38.4	82.8
vGamba-X	224^2	11.3	54	83.2

5.1 Classification

Settings. The ImageNet-1K dataset [6] contains 1.28M training images and 50K validation images across 1,000 categories. Training follows ConvNeXt [21] settings, using augmentations like random cropping, flipping, label-smoothing, mixup, and random erasing. For 224^2 input images, we optimize with AdamW with momentum 0.9, batch size 64, and weight decay 0.05. Then trained vGamba models for 250 epochs with a cosine schedule and EMA, starting with a 1×10^{-3} learning rate.

Results. Table 1 reports ImageNet-1K classification results. Among the three bottleneck designs, vGamba-B outperforms ResNet-50 by +4.9% and BoT-Net by +2.8% in Top-1 accuracy, while reducing FLOPs by 0.1G and 0.3G and parameters by 5.6M and 1.4M respectively, demonstrating a better efficiency–accuracy trade-off among bottleneck designs. Among Transformer-based models, vGamba-B surpasses DeiT-S by +1.3% with 0.9G fewer FLOPs and 3.1M fewer parameters. Among SSM-based models, vGamba-B exceeds ViM-S by +0.8% with 1.6G lower compute. VMamba-T achieves +1.5% higher accuracy than vGamba-B, however it requires 1.2G more FLOPs and 11.1M more parameters, confirming that vGamba-B achieves competitive accuracy without the overhead of multi-directional SSM replication.

5.2 Segmentation

Settings. We conducted semantic segmentation experiments using ADE20K dataset [40] within the UperNet framework [37]. The backbone network was initialized with pretrained weights from ImageNet-1K [6], while the remaining components were randomly initialized. Model optimization was performed using the AdamW optimizer with a batch size of 16. The model was trained for 160k iterations to ensure thorough learning.

Table 2: Comparison of semantic segmentation results on ADE20K using UperNet. FLOPs are measured with an input size of 512×2048.

Backbone	ADE20K Param (M)	FLOP (G)	mIoU (SS)	mIoU (MS)
ResNet-50 [15]	67	953	42.1	42.8
ResNet-101 [15]	85	1030	42.9	44.0
ConvNeXt-T [21]	60	939	46.0	46.7
ConvNeXt-S [21]	82	1027	48.7	49.6
DeiT-S + MLN [32]	58	1217	43.8	45.1
DeiT-B + MLN [32]	144	2007	45.5	47.2
Swin-T [20]	60	945	44.5	45.8
Swin-S [20]	81	1039	47.6	49.5
Swin-B [20]	121	1188	48.1	-
Vim-S [41]	46	–	44.9	–
VMamba-T [19]	62	949	47.9	48.8
VMamba-S [19]	82	1028	50.6	51.2
VMamba-B [19]	122	1170	51.0	51.6
Eff.VMamba-T [25]	14	230	38.9	39.3
Eff.VMamba-S [25]	29	505	41.5	42.1
Eff.VMamba-B [25]	65	930	46.5	47.3
vGamba-B	55	941	46.6	46.7
vGamba-L	71	1019	46.9	47.8

Results. Table 2 reports semantic segmentation results on ADE20K using UperNet. Among CNN backbones, vGamba-B improves over ResNet-50 by +4.5% SS mIoU with 14M fewer parameters and comparable FLOPs, confirming that the Gamba cell enhances spatial context modeling over pure CNN feature extraction. Compared to Transformer-based backbones, vGamba-B surpasses DeiT-S+MLN by +2.8% SS mIoU while requiring 276G fewer FLOPs, and matches Swin-T (+2.1%) at comparable parameters and compute. Among SSM-based models, vGamba-B achieves +1.7% over ViM-S and comparable performance to Eff.VMamba-B (+0.1%) with similar compute. VMamba-T and VMamba-S achieve higher mIoU than vGamba-B by 1.3% and 4.0% respectively, however both require additional parameters and FLOPs, reflecting the cost of multi-directional SSM replication. Overall, vGamba-B achieves a competitive efficiency–accuracy trade-off across all backbone categories.

5.3 Object Detection

Settings. We evaluated our model on object detection and instance segmentation tasks using the COCO 2017 dataset [18], which consists of approximately 118K training images and 5K validation images. vGamba was utilized as the backbone and integrated into the Mask R-CNN framework [14] for feature extraction. The model was initialized with weights pre-trained on ImageNet-1K (300 epochs) and trained for 12 epochs (1×) and 36 epochs (3×).

Results. Table 3 reports object detection and instance segmentation results across three schedules.

Mask R-CNN 1×. vGamba-B achieves comparable detection to VMamba-T (-2.4 AP_b) while requiring 46G fewer FLOPs at similar parameter count, and surpasses both Swin-T and ConvNeXt-T by $+2.2$ and $+0.7$ AP_b respectively with fewer FLOPs, reflecting the efficiency advantage of the single-block SSM design.

Mask R-CNN 3× MS. vGamba-L outperforms VMamba-S by $+1.4$ AP_b and $+1.9$ AP_m with 95G fewer FLOPs and comparable parameters, confirming that vGamba benefits proportionally more from longer training schedules relative to its compute cost.

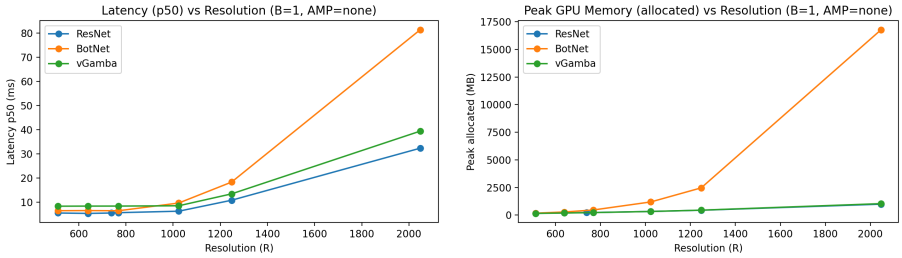
Cascade Mask R-CNN 3× MS. vGamba-B surpasses Swin-T and ConvNeXt-T by $+0.4$ AP_b with 10M fewer parameters at comparable FLOPs. MambaVision-T achieves $+0.2$ AP_b higher than vGamba-B with 10M more parameters, reflecting a competitive efficiency–accuracy trade-off among SSM-based detection backbones.

Table 3: Cascade Mask R-CNN detection results.

Backbone	FLOPs (G)	Params (M)	AP_b	AP_m
Mask R-CNN 1× schedule				
Swin-T [20]	267G	48M	42.7	39.3
ConvNeXt-T [21]	262G	48	44.2	40.1
VMamba-T [19]	271G	50	47.3	42.7
Swin-S [20]	354G	69	44.8	40.9
ConvNeXt-S [21]	348G	70	45.4	41.8
VMamba-S [19]	349G	70	48.7	43.7
Swin-B [20]	496G	107	46.9	42.3
ConvNeXt-B [21]	486G	108	47.0	42.7
VMamba-B [19]	485G	108	49.2	44.1
vGamba-B	225G	49	44.9	42.0
vGamba-L	254G	71	49.8	45.3
Mask R-CNN 3× MS schedule				
Swin-T [20]	267G	48	46.0	41.6
ConvNeXt-T [21]	262G	48	46.2	41.7
VMamba-T [19]	271G	50	48.8	43.7
Swin-S [20]	354G	69	48.2	43.2
ConvNeXt-S [21]	348G	70	47.9	42.9
VMamba-S [19]	349G	70	49.9	44.2
vGamba-L	254G	71	51.3	46.1
Cascade Mask R-CNN 3× MS schedule				
DeiT-Small/16 [32]	889G	80	48.0	41.4
ResNet-50 [15]	739G	82	46.3	40.1
Swin-T [20]	745G	86	50.4	43.7
ConvNeXt-T [21]	741G	86	50.4	43.7
MambaVision-T [13]	740G	86	51.0	44.3
Swin-S [20]	838G	107	51.9	45.0
ConvNeXt-S [21]	827G	108	51.9	45.0
MambaVision-S [13]	828G	108	52.1	45.2
Swin-B [20]	982G	145	51.9	45.0
ConvNeXt-B [21]	964G	146	52.7	45.6
MambaVision-B [13]	964G	145	52.8	45.7
vGamba-B	727G	76	50.8	43.7
vGamba-L	905G	123	52.0	45.3

5.4 Scaling Performance

We evaluate the scaling performance of ResNet-50, BoTNet, and vGamba backbones across input resolutions of 512, 640, 768, 1024, and 2048. For each setting, we measure inference throughput (FPS) and peak GPU memory consumption. This analysis provides insight into each model's computational efficiency and practical suitability under varying resolution and hardware constraints. As



(a) FPS comparison (speed) (b) Memory usage

Fig. 7: Scaling behavior of ResNet-50, BoTNet and vGamba across increasing input resolutions. (a) Throughput decreases more rapidly for BoTNet as spatial size grows. (b) Peak memory usage shows superlinear growth for BoTNet, while vGamba follows a near-linear trend comparable to ResNet-50.

shown in Figure 7, vGamba achieves computational and memory scaling behavior comparable to ResNet-50 [15], indicating near-CNN-level efficiency. In contrast, BoTNet [29] demonstrates substantially higher memory growth and reduced throughput as resolution increases. For example, at 2048×2048 resolution, vGamba is $2.07\times$ faster than BoTNet (39.41 ms vs. 81.47 ms) and reduces peak GPU memory usage by 93.8% (1.03 GB vs. 16.78 GB) under identical inference settings. Additional details are provided in Appendix A.1.

5.5 Qualitative Experiments

To understand model behavior beyond quantitative metrics, we conduct qualitative analyses focusing on interpretability and receptive field characteristics. Specifically, we investigate (1) attention localization using Hi-Res-CAM [9] and (2) effective receptive field (ERF) [22] distributions across layers. These analyses reveal how the proposed architecture captures long-range dependencies and contextual information in complex aerial scenes.

Hi-Res-CAM was applied on the AID dataset [36] (see Appendix A.2 for implementation details) to visualize attention regions. AID contains large-scale aerial scenes with long-range spatial dependencies. As shown in Fig. 8, heatmaps indicate that the model attends to both discriminative local textures and broader semantic layouts, such as runways, roads, and surrounding infrastructure. Unlike CNNs such as ResNet [15], which focus on isolated local patches, ViT-B/16 [8] captures long-range dependencies, though interpretability can be affected by high norms during inference [5] and computation cost is higher. ViM [41] and BotNet [29] also performs competitively. Our approach captures globally coherent regions aligned with scene semantics, confirming that it integrates local and long-range cues and produces meaningful decision regions.

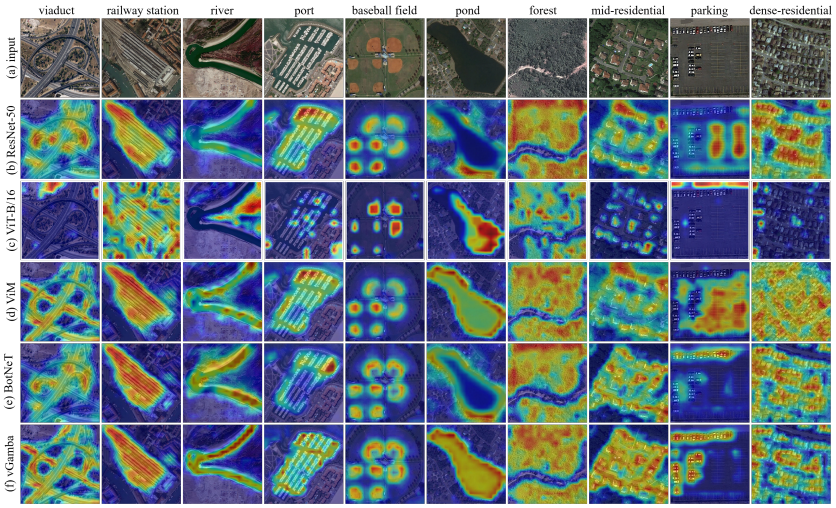


Fig. 8: Hi-Res-CAM visualization on AID, highlighting model decision regions.

ERF quantifies how much of the input image effectively contributes to an output activation, revealing how models capture LRD beyond their theoretical receptive field [22]. We compare ResNet-50, BoTNet, and vGamba under identical settings (see Appendix A.2 for implementation details). As shown in Fig. 9, ResNet-50 exhibits local ERFs, while BoTNet achieves global coverage via attention and vGamba attains comparable global context more efficiently through its SSM-based design. ERFs remain local in layers 0–3 but expand globally in layer 4, validating the Gamba cell placement for efficient LRD modeling.

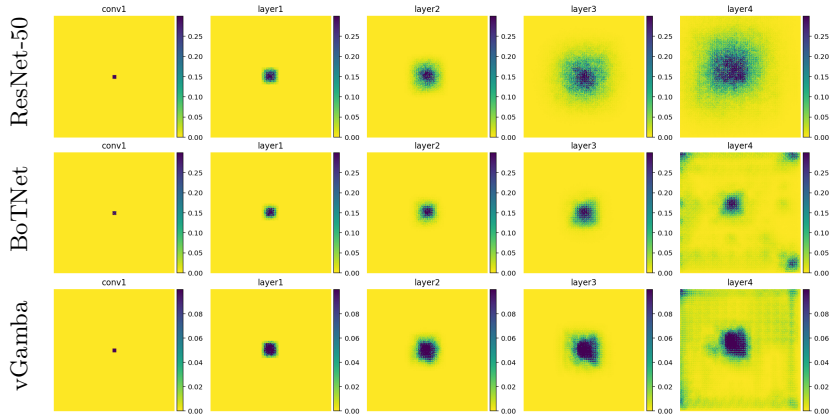


Fig. 9: ERF comparison among ResNet-50, BoTNet, and vGamba across layers.

5.6 Ablation

Effects of Components. Table 4 (a) ablates each GCell component. Adding RPE alone improves top-1 to +2.2% gain without incuring computational complexity, confirming that positional priors compensate for spatial structure lost during flattening. Adding ASC alone results in a stronger +2.6% gain at a marginal cost of +0.5G FLOPs. Combining both achieves the best result at +2.8%, showing that RPE and ASC are complementary rather than redundant.
Effects of Attention. Table 4 (b) compares ASC against established attention alternatives under identical settings. SE Block [17] and CBAM [34] underperform by −1.1% and −0.8% respectively, SE operates channel-wise only, while CBAM lacks directional bias. Coordinate Attention [16] falls short by −0.5%, lacking per-channel gating and axis-wise biases for asymmetric spatial priors. ASC achieves the highest accuracy at comparable FLOPs and parameters, confirming that per-channel gating and axis-wise biases are the key differentiators. See Appendix A.3 for additional ablations.

Table 4: Ablation (a) Gamba cell components and (b) spatial attention mechanisms.

(a) Components	Top-1	FLOPs	Param
None	78.3%	3.38G	19.38M
RPE *only*	80.5%	3.38G	19.38M
ASC *only*	80.9%	3.88G	19.45M
RPE + ASC	81.1%	3.88G	19.45M

(b) Attention	Top-1	FLOPs	Param
SE Block [17]	80.0%	3.87G	19.47M
CBAM [34]	80.3%	3.87G	19.47M
Coord. Attn. [16]	80.6%	3.88G	19.45M
ASC (Ours)	81.1%	3.88G	19.45M

6 Conclusion

This work advances bottleneck design for LRD modeling by demonstrating that a single, lightweight SSM-enhanced bottleneck can replace both CNN and Transformer bottlenecks while efficiently capturing global context. The Gamba bottleneck embeds a spatially aware SSM (the Gamba cell) and the ASC module into a compact structure that maintains representational strength. The resulting backbone, vGamba, outperforms CNN- and Transformer-based alternatives across downstream tasks, capturing LRD more accurately and with smoother convergence while using lower memory and FLOPs. Overall, vGamba provides a practical, scalable alternative to existing visual backbones.

Limitations. While vGamba demonstrates strong performance across standard benchmarks, its effectiveness on domain-specific tasks requiring extreme precision, such as medical imaging, remains unevaluated. Furthermore, restricting the Gamba cell to stage 4 only may limit fine-grained feature modulation in earlier stages, which could affect performance on dense prediction tasks such as segmentation, as reflected in the segmentation results (Sec. 5.2).

References

1. Chen, C.F.R., Fan, Q., Panda, R.: Crossvit: Cross-attention multi-scale vision transformer for image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 357–366 (2021)
2. Chen, X., Liu, Z., Tang, H., Yi, L., Zhao, H., Han, S.: Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2061–2070 (2023)
3. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
4. Dai, Z., Liu, H., Le, Q.V., Tan, M.: Coatnet: Marrying convolution and attention for all data sizes. In: Advances in Neural Information Processing Systems. vol. 34, pp. 3965–3977 (2021)
5. Darcet, T., Oquab, M., Mairal, J., Bojanowski, P.: Vision transformers need registers. arXiv preprint arXiv:2309.16588 (2023)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., 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)
7. Ding, X., Zhang, X., Han, J., 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 (2022)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
9. Draelos, R.L., Carin, L.: Use hirescam instead of grad-cam for faithful explanations of convolutional neural networks. arXiv preprint arXiv:2011.08891 (2020)
10. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
11. Gu, A., Goel, K., Ré, C.: Efficiently modeling long sequences with structured state spaces. In: International Conference on Learning Representations (2022), `https://arxiv.org/abs/2111.00396`
12. Haruna, Y., Qin, S., Chukkol, A.H.A., Yusuf, A.A., Bello, I., Lawan, A.: Exploring the synergies of hybrid convolutional neural network and vision transformer architectures for computer vision: A survey. Engineering Applications of Artificial Intelligence **144**, 110057 (2025). `https://doi.org/10.1016/j.engappai.2025.110057`
13. Hatamizadeh, A., Kautz, J.: Mambavision: A hybrid mamba-transformer vision backbone. arXiv preprint arXiv:2407.08083 (2024)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2961–2969 (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. Hou, Q., Zhou, D., Feng, J.: Coordinate attention for efficient mobile network design. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13713–13722 (2021)
17. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)

18. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V. pp. 740–755. Springer International Publishing (2014)
19. Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., Jiao, J., Liu, Y.: Vmamba: Visual state space model. Advances in neural information processing systems **37**, 103031–103063 (2025)
20. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., 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)
21. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11976–11986 (2022)
22. Luo, W., Li, Y., Urtasun, R., Zemel, R.: Understanding the effective receptive field in deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 29 (2016)
23. Ma, X., Kong, X., Wang, S., Zhou, C., May, J., Ma, H., Zettlemoyer, L.: Luna: Linear unified nested attention. vol. 34, pp. 2441–2453 (2021)
24. Nguyen, E., Goel, K., Gu, A., Downs, G., Shah, P., Dao, T., Baccus, S., Ré, C.: S4nd: Modeling images and videos as multidimensional signals with state spaces. In: Advances in Neural Information Processing Systems. vol. 35, pp. 2846–2861 (2022)
25. Pei, X., Huang, T., Xu, C.: Efficientvmamba: Atrous selective scan for light weight visual mamba. arXiv preprint arXiv:2403.09977 (2024)
26. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters – improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4353–4361 (2017)
27. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10428–10436 (2020)
28. Song, J.g.: Ufo-vit: High performance linear vision transformer without softmax. arXiv preprint arXiv:2109.14382 (2021)
29. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16519–16529 (2021)
30. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
31. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114 (2019)
32. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021)
33. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7794–7803 (2018)
34. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: CBAM: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision. pp. 3–19 (2018)

35. Wu, K., Peng, H., Chen, M., Fu, J., Chao, H.: Rethinking and improving relative position encoding for vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10033–10041 (2021)
36. Xia, G.S., Hu, J., Hu, F., Shi, B., Bai, X., Zhong, Y., Zhang, L., Lu, X.: Aid: A benchmark data set for performance evaluation of aerial scene classification. IEEE Transactions on Geoscience and Remote Sensing **55**(7), 3965–3981 (2017)
37. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 418–434 (2018)
38. Yang, Y., Chiang, H.Y., Li, G., Marculescu, D., Marculescu, R.: Efficient low-rank backpropagation for vision transformer adaptation. vol. 36 (2024)
39. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 472–480 (2017)
40. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. International Journal of Computer Vision **127**, 302–321 (2019)
41. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision mamba: Efficient visual representation learning with bidirectional state space model. arXiv (2024), https://arxiv.org/abs/2401.09417