MobileMamba: Lightweight Multi-Receptive Visual Mamba Network

Haoyang He[1*] Jiangning Zhang[1,2*] Yuxuan Cai[3] Hongxu Chen[1] Xiaobin Hu[2]
Zhenye Gan[2] Yabiao Wang[1,2] Chengjie Wang[2] Yunsheng Wu[2] Lei Xie[1†]
[1]Zhejiang University [2]Youtu Lab, Tencent [3]Huazhong University of Science and Technology
Code: https://github.com/lewandofskee/MobileMamba

Abstract

Previous research on lightweight models has primarily focused on CNNs and Transformer-based designs. CNNs, with their local receptive fields, struggle to capture long-range dependencies, while Transformers, despite their global modeling capabilities, are limited by quadratic computational complexity in high-resolution scenarios. Recently, state-space models have gained popularity in the visual domain due to their linear computational complexity. Despite their low FLOPs, current lightweight Mamba-based models exhibit suboptimal throughput. In this work, we propose the ***MobileMamba*** *framework, which balances efficiency and performance. We design a three-stage network to enhance inference speed significantly. At a fine-grained level, we introduce the* ***M****ulti-****R****eceptive* ***F****ield* ***F****eature* ***I****nteraction (MRFFI) module, comprising the Long-Range* ***W****avelet* ***T****ransform-****E****nhanced* ***Mamba*** *(WTE-Mamba), Efficient* ***M****ulti-****K****ernel* ***De****pthwise* ***Conv****olution (MK-DeConv), and Eliminate Redundant* ***Identity*** *components. This module integrates multi-receptive field information and enhances high-frequency detail extraction. Additionally, we employ training and testing strategies to further improve performance and efficiency. MobileMamba achieves up to* ***83.6%*** *on Top-1, surpassing existing state-of-the-art methods which is maximum* $\times$***21***$\uparrow$ *faster than LocalVim on GPU. Extensive experiments on high-resolution downstream tasks demonstrate that MobileMamba surpasses current efficient models, achieving an optimal balance between speed and accuracy.*

1. Introduction

The proliferation of mobile devices has increased the demand for efficient and accurate visual processing in resource-constrained environments. Lightweight models significantly reduce computational and storage costs while enhancing inference speed. Current lightweight models

[*]Equal contributions.
[†]Corresponding author.

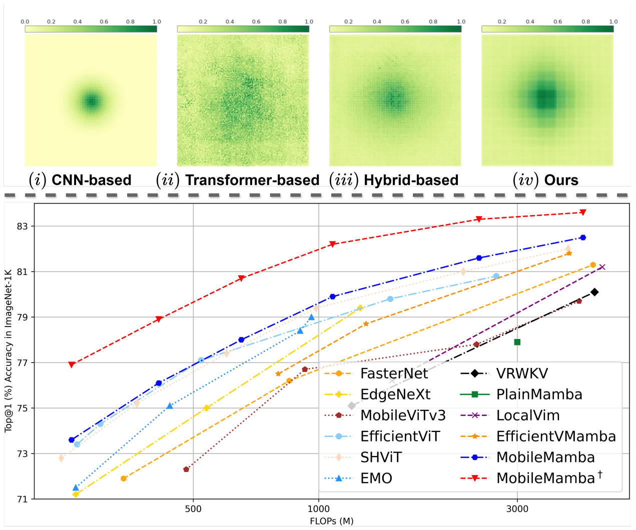


Figure 1. **Top**: Visualization of the *Effective Receptive Fields* (ERF) for different architectures. **Bottom**: Performance *vs.* FLOPs with recent CNN/Transformer/Mamba-based methods.

are primarily categorized into CNN-based and Transformer-based structures. CNN-based MobileNets [24, 26, 54] use depth-wise separable convolutions to reduce computational complexity, laying a foundation for subsequent CNN-based work [3, 44, 59, 61, 77]. However, the major drawback of CNN-based methods is their local **E**ffective **R**eceptive **F**ield (ERF), as shown in Fig. 1*(i)*, which is *confined to the central region and lacks long-range correlations.* In downstream tasks (Tab. 5) with high-resolution inputs, CNN-based methods can only achieve performance improvements by increasing computational load.

Vision **T**ransformers (ViTs) exhibit a global ERF and long-range modeling capabilities in Fig. 1*(ii)*. However, their quadratic computational complexity results in higher overhead compared to CNNs. Some works [32, 33, 37, 45, 54, 65, 74] have reduced resolution or channel count to alleviate this complexity, achieving notable results. Despite this, pure ViTs lack inductive bias, prompting researchers to develop hybrid CNN-ViT structures [31, 72, 76] that combine local and global ERF for improved performance in

Fig. 1*(iii)*. However, *ViT-based methods still face the issue of quadratic computational complexity, especially with high-resolution inputs in downstream tasks (Tab. 6).*

State-space models [13–15, 56] have gained attention for capturing long-range dependencies with linear computational complexity. Researchers have successfully applied these models to the visual domain [38, 55, 81], achieving notable effectiveness and efficiency. The recent lightweight Mamba-based models [28, 49] introduce different efficient scanning methods to reduce complexity. However, only FLOPs are reported in their works, which do not necessarily correlate with fast inference speed. Experimental results in Fig. 2 show that current Mamba-based structures suffer from slow inference speeds and poor performance.

Based on the above motivation, we propose **MobileMamba**, designed as an efficient lightweight network through *Coarse-Grained*, *Fine-Grained*, and *Training/Testing Strategies*. Firstly, in Sec. 3.1, we discuss the trade-offs between four-stage and three-stage networks in terms of accuracy, speed, and FLOPs. As shown in Fig. 3, under the same throughput, a three-stage network achieves higher accuracy. Similarly, for the same performance, a three-stage network has higher throughput. Therefore, we select a three-stage network as our Coarse-Grained framework. In the design of the MobileMamba module in Sec. 3.2, we introduce an efficient **M**ulti-**R**eceptive **F**ield **F**eature **I**nteraction (MRFFI) module. Specifically, the input features are divided into three parts along the channel dimension. The first part uses a Long-Range ***W**avelet **T**ransform-**E**nhanced Mamba* (WTE-Mamba) module to extract global features while enhancing the extraction of fine-grained details such as edge information. The second part employs ***M**ulti-**K**ernel **De**pthwise Convolution* (MK-DeConv) operations to capture multi-scale receptive fields. The final part uses *Eliminate redundant **Identity*** mapping to reduce channel redundancy in high-dimensional space, decreasing computational complexity and increasing processing speed. The features obtained through *MRFFI integrate global and multi-scale local receptive field information, enhancing the extraction of high-frequency edge details.* Finally, we enhance the model's learning capability through two training phase strategies in Sec. 3.3, Knowledge Distillation, and Extended Training Epochs. Additionally, a Normalization Layer Fusion strategy in the testing phase improves the model's inference speed.

In Fig. 1*(iv)*, our approach utilizes a global ERF, whereas multi-kernel local convolution operations facilitate the extraction of adjacent information. The comparison with SoTA methods at the bottom of Fig. 1 shows that MobileMamba† (with training strategies) achieves Top-1 accuracies of 76.9/78.9/80.7/82.2/83.3/83.6 on ImageNet-1K [10] for models ranging from 200M to 4G FLOPs, surpassing existing CNN, ViT, and Mamba-based methods. Compared to efficient Mamba-based methods in Fig. 2, MobileMamba improves Top-1 by +0.7↑ while being ×**21**↑ times faster than LocalVim [29], and improves by +2.0↑ while being ×**3.3**↑ times faster than EfficientVMamba [49]. This demonstrates a significant advantage over existing Mamba-based lightweight model designs. Extensive experiments on downstream tasks further validate the effectiveness of our method. On Mask RCNN [22], MobileMamba improves mAP^b by +1.3↑, mAP^m by +1.0↑ and throughput by +56%↑ compared to EMO [76]. On RetinaNet [34], it improves mAP^b by +2.1↑ and throughput by ×**4.3**↑ compared to EfficientVMamba [49]. On SSDLite [25], it achieves mAPs of 24.0/29.5 by increasing resolution. On DeepLabv3 [5], Semantic FPN [30], and PSPNet [78], it achieves mIoUs of up to 37.4/42.5/36.9 with fewer FLOPs. Compared to CNN-based MobileNetv2 [54] and ViT-based MobileViTv2 [46], our approach achieves improvements of +7.2↑ and +0.4↑, respectively, in high-resolution 512x512 input downstream tasks, while only requiring **8.5%** and **11.2%** of their FLOPs for PSPNet [78].

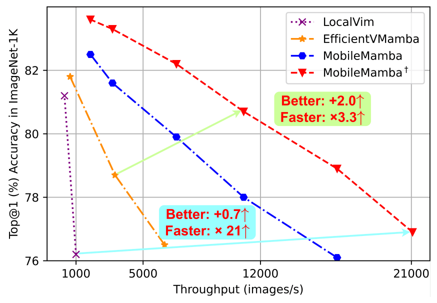


Figure 2. **Accuracy *vs.* Speed** with Mamba-based methods.

In summary, our contributions are as follows:

- We propose a lightweight three-stage MobileMamba framework that achieves a good balance between performance and efficiency. The effectiveness and efficiency of MobileMamba have been validated on classification tasks as well as three high-resolution input downstream tasks.
- We designed an efficient Multi-Receptive Field Feature Interaction (MRFFI) module to enhance multi-scale perception capabilities with larger ERF and improve the extraction of fine-grained high-frequency edge information.
- MobileMamba significantly enhances performance and efficiency by employing training and testing strategies across a range of models of different FLOPs sizes.

2. Related Work

2.1. Lightweight Visual Models

The most extensively studied lightweight visual networks can be categorized into CNN-based and Vision Transformer (ViT)-based structures. The CNN-based MobileNets [24, 26, 54] transitions from standard convolution to depthwise separable convolution, significantly reducing computational complexity. GhostNets [18, 39, 61] replaces the original convolution with a cheap operation on half of the channels. Additionally, numerous CNN-based works [3, 43, 59, 60, 77]demonstrate excellent performance and efficiency on mobile devices. The main limitation of these methods is their local receptive field. In contrast, ViT possesses a global receptive field and the ability to capture long-range dependencies. Nevertheless, their quadratic computational complexity results in higher computational costs compared to CNNs. Therefore, lightweight vision Transformers are designed to retain their global receptive field while reducing computational overhead. EfficientViT [37] designs a three-stage network and proposes Cascaded Group Attention to significantly improve inference speed. SHViT [74] introduces Single Head Self-Attention, selecting only a few channels to use ViT while directly connecting the remaining channels via Identity, greatly enhancing operational efficiency. Furthermore, many hybrid methods [32, 33, 45, 46, 48, 63, 65, 76], have achieved outstanding performance.

2.2. State Space Models

State Space Models (SSMs) [13, 15–17, 56] inspired by control systems, can be regarded as linear time-invariant systems mapping input $x(t) \in \mathbb{R}^L$ to output $y(t) \in \mathbb{R}^L$ via hidden state $h(t) \in \mathbb{R}^M$: $h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), y(t) = \mathbf{C}h(t)$, where $\mathbf{A} \in \mathbb{R}^{M\times M}$, $\mathbf{B} \in \mathbb{R}^{M\times 1}$, and $\mathbf{C} \in \mathbb{R}^{1\times M}$.

Mamba [14] uses zero-order hold with timescale Δ to convert continuous $\mathbf{A}$ and $\mathbf{B}$ to discrete $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$:

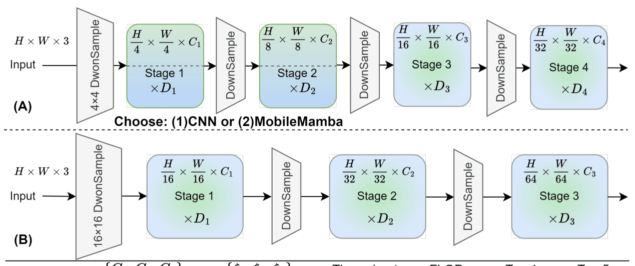
$$\overline{\mathbf{A}} = \exp(\Delta\mathbf{A}), \quad \overline{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}. \tag{1}$$

The discrete system is: $h_t = \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, y_t = \mathbf{C}h_t$. From a global convolution perspective:

$$\overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}}\overline{\mathbf{B}}, \dots, \mathbf{C}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}), \quad \mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}, \tag{2}$$

where $*$ is convolution, L is the sequence length, and $\overline{\mathbf{K}} \in \mathbb{R}^L$ is the SSM kernel.

SSMs for Vision. SSMs [13, 15, 56] have garnered significant attention due to their efficient computational complexity in capturing long-range dependencies. Mamba [14] introduces the S6 module, achieving a simple structure with excellent efficiency in long-sequence modeling. Due to this advantage, numerous works have applied it to visual tasks [19, 20, 38, 42, 52, 70, 81]. Vim [81] proposes a bidirectional Mamba block, demonstrating its speed and memory advantages over ViTs at high resolutions. VMamba [38] introduces Cross-Scan to enhance modeling capabilities. EfficientVMamba [49] proposes Efficient Scan, improving scanning efficiency through skip sampling. LocalVim [28] proposes local window scanning to enhance local information acquisition. Despite various designs, no lightweight Mamba-based network surpasses existing CNN and ViT methods. This paper explores lightweight Mamba-based visual networks to achieve better performance, lower computational complexity, and faster inference speed.


$H \times W \times 3$
Input
4×4 DwonSample
$\frac{H}{4} \times \frac{W}{4} \times C_1$
Stage 1
$\times D_1$
DownSample
$\frac{H}{8} \times \frac{W}{8} \times C_2$
Stage 2
$\times D_2$
DownSample
$\frac{H}{16} \times \frac{W}{16} \times C_3$
Stage 3
$\times D_3$
DownSample
$\frac{H}{32} \times \frac{W}{32} \times C_4$
Stage 4
$\times D_4$
(A)
Choose: (1)CNN or (2)MobileMamba
$H \times W \times 3$
Input
16×16 DwonSample
$\frac{H}{16} \times \frac{W}{16} \times C_1$
Stage 1
$\times D_1$
DownSample
$\frac{H}{32} \times \frac{W}{32} \times C_2$
Stage 2
$\times D_2$
DownSample
$\frac{H}{64} \times \frac{W}{64} \times C_3$
Stage 3
$\times D_3$
(B)

Architecture	$\{C_1, C_2, C_3\}$ $\{D_1, D_2, D_3\}$	$\{\xi_1, \xi_2, \xi_3\}$ $\{\mu_1, \mu_2, \mu_3\}$	Throughput (images/s)	FLOPs (M)	Top-1 (%)	Top-5 (%)
(A)(1)	{48,136,320,400} {2,2,3,3}	{0.0,0.0,0.2,0.2} {0.8,0.8,0.6,0.6}	10611	499	76.9	93.2
(A)(1)	{72,144,336,432} {1,1,3,3}	{0.0,0.0,0.2,0.2} {0.8,0.8,0.6,0.6}	10684	557	77.6	93.6
(A)(2)	{48,144,336,384} {1,1,3,4}	{0.2,0.2,0.4,0.4} {0.6,0.5,0.4,0.3}	10553	517	77.4	93.5
(B)	{192, 384, 448} {1,2,2}	{0.8,0.7,0.6} {0.2,0.2,0.3}	**11000**	652	**78.0**	**94.0**

Figure 3. **Coarse-Grained Design**. (A) illustrates the structure of a commonly used four-stage network, where the first two stages can be configured with either (1) a purely CNN-based structure or (2) the MobileMamba structure. (B) depicts the three-stage network structure employed in this study. The following table presents the model parameters for different structures and the ImageNet-1K Top-1 and Top-5 at equivalent throughput.

3. Methdology

3.1. Coarse-Grained Design of MobileMamba

In this section, we design the efficient MobileMamba structure, which includes a three-stage network as shown in Fig. 3(B). Most existing network [3, 32, 72] follow the four-stage framework depicted in Fig. 3(A). Specifically, in a four-stage network, the first downsampling reduces the input image $H \times W \times 3$ to $\frac{H}{4} \times \frac{W}{4} \times C_1$, and the final output feature map is $\frac{H}{32} \times \frac{W}{32} \times C_4$. In contrast, the three-stage network reduces the input image to $\frac{H}{16} \times \frac{W}{16} \times C_1$ during the first downsampling, and the final output feature map is $\frac{H}{64} \times \frac{W}{64} \times C_4$. Due to the larger feature map size in the four-stage network, it requires more computation and consequently operates at a slower speed. The table below Fig. 3 compares the classification results on the ImageNet-1K [10] dataset for the three-stage network and various four-stage networks under similar throughput conditions. In the first two experiments, the initial two stages of the four-stage network are designed with a purely CNN architecture, which enhances inference speed. The third experiment employs the MobileMamba blocks across all four stages of the network. The results indicate that although the four-stage network with a purely CNN structure in the first two stages shows improved inference speed and performance, *the three-stage network achieves faster inference with both Top-1 and Top-5 accuracy improvement of +0.4*↑. Ultimately, we select the three-stage network structure to enhance inference speed and improve classification results.

3.2. Fine-Grained Design of MobileMamba

The proposed efficient **M**ulti-**R**eceptive **F**ield **F**eature **I**nteraction (**MRFFI**) module is placed between symmet-

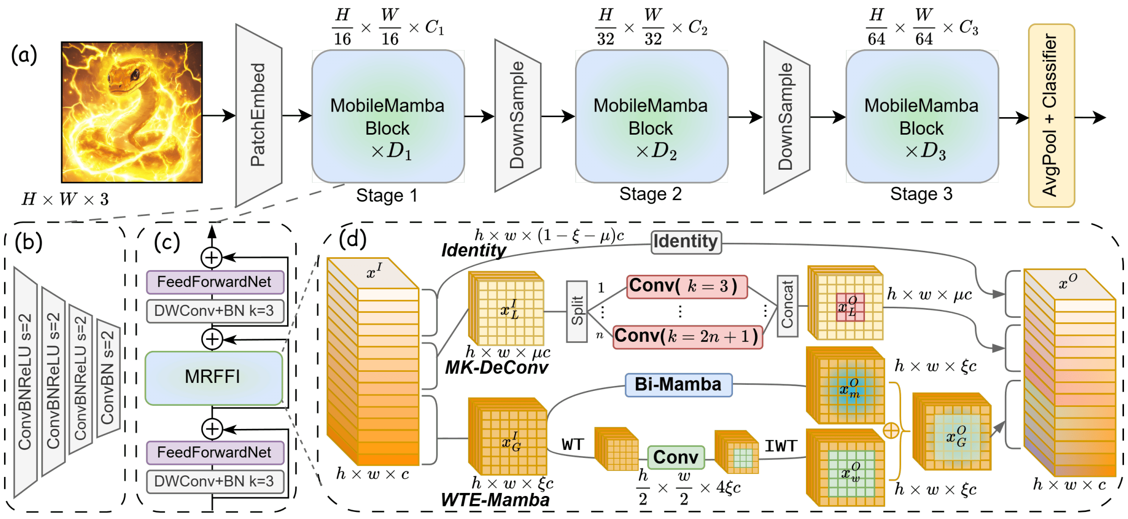

(a)
$H \times W \times 3$
PatchEmbed
$\frac{H}{16} \times \frac{W}{16} \times C_1$
MobileMamba Block $\times D_1$
Stage 1
DownSample
$\frac{H}{32} \times \frac{W}{32} \times C_2$
MobileMamba Block $\times D_2$
Stage 2
DownSample
$\frac{H}{64} \times \frac{W}{64} \times C_3$
MobileMamba Block $\times D_3$
Stage 3
AvgPool + Classifier
(b)
ConvBNReLU s=2
ConvBNReLU s=2
ConvBNReLU s=2
ConvBN s=2
(c)
FeedForwardNet
DWConv+BN k=3
MRFFI
FeedForwardNet
DWConv+BN k=3
(d)
x^I
$h \times w \times c$
Identity
$h \times w \times (1-\xi-\mu)c$
Identity
x^I_L
$h \times w \times \mu c$
MK-DeConv
Split
Conv($k=3$)
Conv($k=2n+1$)
Concat
x^O_L
$h \times w \times \mu c$
Bi-Mamba
x^O_m
$h \times w \times \xi c$
x^I_G
$h \times w \times \xi c$
WTE-Mamba
WT
Conv
$\frac{h}{2} \times \frac{w}{2} \times 4\xi c$
IWT
x^O_w
$h \times w \times \xi c$
x^O_G
x^O
$h \times w \times c$

Figure 4. **Overview of MobileMamba**. (a) Architecture of MobileMamba. (b) 16 ×16 DownSample PatchEmbed. (c) Structure of MobileMamba Block. (d) **Fine-Grained Design.** The proposed efficient Multi-Receptive Field Feature Interaction (MRFFI) module.

ric local information perception and an FFN in each MobileMamba block. In the MRFFI module, features are divided into three parts along the channel dimension. *1)*. The first part of the features undergoes the Long-Range ***Wavelet Transform-Enhanced Mamba*** *(WTE-Mamba)*, which enhances the extraction of high-frequency edge information while performing global modeling. *2)*. The second part is processed through ***Multi-Kernel Depthwise Convolution*** *(MK-DeConv)* operations to enhance the perception capability of different receptive fields. *3)*. The remaining features are subjected to *Identity* mapping to reduce feature redundancy in high-dimensional space and decrease computational complexity, thereby improving processing speed.

Long-range WTE-Mamba. The purpose is to enhance the ability to extract *fine-grained information*, such as *high-frequency edge details, based on global modeling*. Also, the convolution operations [12] on the WT feature maps have a *larger ERF* compared to normal scales and exhibit *lower computational complexity*. For the input feature $x^I \in \mathbb{R}^{h \times w \times c}$, the features $x^I_G \in \mathbb{R}^{h \times w \times \xi c}$ are processed through a bidirectional scanning Mamba module to learn global information, with the global channel proportion denoted as $0 \leq \xi \leq 1$.

$$\begin{aligned} x^I_{m1} &= \mathrm{SSM}\left(\sigma\left(\mathrm{Conv}\left(\mathrm{Linear}\left(x^I_G\right)[:\xi c]\right)\right)\right), \\ x^I_{m2} &= \sigma\left(\mathrm{Linear}\left(x^I_G\right)[\xi c:]\right), \\ x^O_m &= \mathrm{Linear}\left(x^I_{m1} \otimes x^I_{m2}\right). \end{aligned} \tag{3}$$

Simultaneously, the same feature map undergoes Haar Wavelet transformation to obtain feature representations $x^I_w \in \mathbb{R}^{\frac{h}{2} \times \frac{w}{2} \times 4\xi c}$ at different frequency scales. Local convolutional information extraction and Inverse Wavelet Transformation (IWT) are then performed to restore the original feature map size $x^O_w \in \mathbb{R}^{h \times w \times \xi c}$.

$$\begin{aligned} x^I_{wt} &= \mathrm{WT}\left(x^I_w, [f_{LL}, f_{LH}, f_{HL}, f_{HH}]\right), \\ x^O_w &= \mathrm{IWT}\left(\mathrm{Conv}(x^I_{wt}), [f_{LL}, f_{LH}, f_{HL}, f_{HH}]\right), \end{aligned} \tag{4}$$

where $f_{LL} = \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$ is a low-pass filter, and $f_{LH} = \frac{1}{2}\begin{bmatrix}1 & -1\\1 & -1\end{bmatrix}$, $f_{HL} = \frac{1}{2}\begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix}$, $f_{HH} = \frac{1}{2}\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}$ are a set of high filters. The final output feature map for this part is obtained by adding the output feature map from the Mamba module, which has extracted global information, to the output feature map from the wavelet-transformed and convolved local information.

$$x^O_G = x^O_m + x^O_w, \text{where } x^O_G \in \mathbb{R}^{h \times w \times \xi c}, \tag{5}$$

Efficient MK-DeConv. This approach extracts local information with *varying ERF*, enabling *multi-receptive field interaction*. For the remaining features, $x^I_L \in \mathbb{R}^{h \times w \times \mu c}$ are selected, where the local channel proportion is denoted as $\mu \leq 1 - \xi$. These channels are then divided into $n \in \mathbb{N}$ parts. Each part $x^I_{Lj} \in \mathbb{R}^{h \times w \times \frac{\mu c}{n}}$ undergoes local convolution operations with different kernel sizes. Finally, the results from the different convolution operations are concatenated to form the output features $x^O_L \in \mathbb{R}^{h \times w \times \mu c}$.

$$\begin{aligned} x^O_{Lj} &= \mathrm{Conv}\left(x^I_{Lj}, \mathrm{k} = (2j+1)\right), j \in 1, ..., n. \\ x^O_L &= \mathrm{Concat}([x^O_{L1}, ..., x^O_{Ln}], \mathrm{dim} = -1), \end{aligned} \tag{6}$$

Eliminate redundant Identity. Finally, to *reduce the issue of feature redundancy in high-dimensional space* [18], we apply identity mapping to the remaining $(1-\xi-\mu)c$ channels. This approach minimizes unnecessary computations and enhances operational efficiency. Therefore, the final output after processing through the MRFFI module is computed as follows:

$$x^O = \mathrm{Concat}(x^O_G, x^O_L, x^I[(1-\xi-\mu)c:]). \tag{7}$$

Model	Reso.	$\{C_1, C_2, C_3\}$	$\{D_1, D_2, D_3\}$	$\{\xi_1, \xi_2, \xi_3\}$	$\{\mu_1, \mu_2, \mu_3\}$
MobileMamba-T2	192	{144, 272, 368}	{1, 2, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}
MobileMamba-T4	192	{176, 368, 448}	{1, 2, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}
MobileMamba-S6	224	{192, 384, 448}	{1, 2, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}
MobileMamba-B1	256	{200, 376, 448}	{2, 3, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}
MobileMamba-B2	384	{200, 376, 448}	{2, 3, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}
MobileMamba-B4	512	{200, 376, 448}	{2, 3, 2}	{0.8, 0.7, 0.6}	{0.2, 0.2, 0.3}

Table 2. **Architecture details** of MobileMamba model variants.

MobileMamba is designed with six structures in Tab. 2. Across different models, we maintain the same global and local channel proportions. For the small model, we use a *smaller input resolution to achieve lower computational complexity and faster runtime*. Conversely, for the large model, we use a *larger input resolution to obtain better performance*, as detailed in Sec. 4.4. By adjusting the input resolution according to the model size, we balance the trade-offs between computational efficiency and performance. This design strategy ensures that MobileMamba can be effectively scaled to meet different requirements while maintaining consistent channel proportions.

Method	FLOPs	Throughput	Params	Top-1
TTT [57]	625	9569	14.2	77.0
xLSTM [2]	695	6868	14.6	77.3
RWKV6 [50]	658	10331	14.8	77.8
Mamba [14]	652	**11000**	15.0	**78.0**

Table 1. Other RNN Paradigm models.

Replace Mamba with other RNN paradigm models. We replace Mamba with the currently popular global ERF RNN paradigm models that have linear computational complexity. The results are shown in Tab. 1. Under similar FLOPs, *Mamba [14] still demonstrates the best performance and higher throughput.* RWKV6 [50] achieves results second only to Mamba. In contrast, TTT [57] and xLSTM [2] fall short in both throughput and performance compared to the former.

3.3. Training and Testing Strategies

We employ two training strategies to further enhance the performance and efficiency of the small model while maintaining the same number of parameters and computational complexity. Additionally, we use a testing strategy to ensure model effectiveness while improving inference speed.

Knowledge Distillation. To enable the lightweight student model, MobileMamba, to learn from the more robust teacher classification model, we follow the Soft Distillation setup from DeiT [62]. This involves minimizing the Kullback-Leibler divergence between the softmax outputs of the teacher model and the student model.

Extended Training Epochs. We observe that under the conventional 300 epochs, the loss of the small model MobileMamba has not fully converged, and the Top-1 accuracy has not reached its potential. Therefore, to improve the performance ceiling of the lightweight model, we extend the training to 1000 epochs.

Normalization Layer Fusion. Convolution operations are typically followed by batch normalization. During inference, batch normalization can be fused with preceding convolution or linear layers. Recalculating the new convolution layer's weights and biases ensures its combined output matches the original layers' output. This fusion enhances computational efficiency and speeds up the forward pass by reducing the number of layers.

4. Experiments

4.1. Implementation Details

We conduct image classification on the ImageNet-1K [10] dataset. The baseline model is trained from scratch for 300 epochs at a resolution of 224^2. The AdamW [40] optimizer is employed with betas (0.9, 0.999), a weight decay of 5e-2, a learning rate of 1.5e-3, and a batch size of 1024. We use a Cosine scheduler [41] with 20 warmup epochs, Label Smoothing [58] of 0.1, stochastic depth [27], and RandAugment [9] during training. For a fair comparison, we follow the same data augmentation techniques proposed in [62], including Mixup [75], random erasing [79], and auto-augmentation [8]. For the enhanced model†, we train for 1000 epochs and follow the Knowledge Distillation recipe as used in DeiT [62] with TResNet-L [51] as the teacher model. For object detection tasks, we validate using lightweight SSDLite [25] and RetinaNet [34] on the MS-COCO 2017 [35] dataset. For instance segmentation tasks, we conduct experiments using Mask R-CNN [22] on the COCO [35] dataset. For semantic segmentation tasks, we experiment with DeepLabv3 [5], PSPNet [78], and FPN [36] on the ADE20K [80] dataset. For all downstream task experiments, we use the standard MMDetection [4] and MMSegmentation [7] libraries, and we only replace the optimizer with AdamW [40] without tuning other parameters. GPU throughput is measured on a single Nvidia L40S Cloud Server with a batch size of 256.

4.2. MobileMamba on ImageNet-1K Classification

The results of MobileMamba across six different model scales compared to other SoTA methods on ImageNet-1K are presented in Tab. 3. The different model scales are categorized by FLOPs. For instance, compared to the MobileMamba-B1 model, the B2 and B4 models only increase the input resolution without adding network depth or width. In the first two model scales, there are currently no Mamba-based models with equivalent FLOPs. MobileMamba-T2 outperforms Transformer-based SHViT-S1 [74] by +0.8$\uparrow$ in Top-1. MobileMamba-T4 surpasses the linear attention-based VRWKV-T [11] by +1$\uparrow$ in Top-1 while having only 33% of its FLOPs. For the MobileMamba-S6 and B1 models, we also observe significant improvements over other CNN, Transformer, and Mamba-based models. MobileMamba-S6 achieves 1.5 higher Top-1 accuracy than EfficientVMamba-T [49] while reducing FLOPs by 18.5%$\downarrow$.

Model	TP↑	FLOPs↓	Params↓	Reso.	Top-1	#Pub
EdgeNeXt-XXS [44]	10532	260	1.3	224	71.2	ECCVW'22
ShuffleNetV2×1.5 [43]	10598	300	3.5	224	72.6	ECCV'18
FasterNet-T0 [3]	19517	340	3.9	224	71.9	CVPR'23
MobileOne-S0 [64]	15805	275	2.1	224	71.4	CVPR'23
MobileViTv3-0.5 [65]	9302	481	1.4	256	72.3	arXiv'2209
EfficientViT-M2 [37]	18693	201	4.2	224	70.8	CVPR'23
EMO-1M [76]	8361	261	1.3	224	71.5	ICCV'23
SHViT-S1 [74]	21516	241	6.3	224	72.8	CVPR'24
MobileMamba-T2	21071	255	8.8	**192**	**73.6**	CVPR'25
MobileMamba-T2†	21071	255	8.8	**192**	**76.9**	CVPR'25
EdgeNeXt-XS [44]	6423	540	2.3	224	75.0	ECCVW'22
InceptionNeXt-A [73]	9238	510	4.2	224	75.3	CVPR'24
MobileOne-S1 [64]	12552	825	4.8	224	75.9	CVPR'23
EfficientFormerV2-S0 [33]	1324	400	3.5	224	75.7	ICCV'23
EfficientViT-M4 [37]	14612	299	8.8	224	74.3	CVPR'23
EMO-2M [76]	6301	439	2.3	224	75.1	ICCV'23
SHViT-S2 [74]	17816	366	11.4	224	75.2	CVPR'24
VRWKV-T [11]	3993	1200	6.2	224	75.1	ICLR'25
MobileMamba-T4	16571	413	14.2	**192**	**76.1**	CVPR'25
MobileMamba-T4†	16571	413	14.2	**192**	**78.9**	CVPR'25
FasterNet-T1 [3]	12820	850	7.6	224	76.2	CVPR'23
MobileOne-S2 [64]	6725	1299	7.8	224	77.4	CVPR'23
ConvNeXtV2-F [69]	5238	780	5.2	224	78.0	CVPR'23
MobileViTv3-XS [65]	3521	927	2.5	256	76.7	arXiv'2209
EfficientViT-M5 [37]	10271	522	12.4	224	77.1	CVPR'23
SHViT-S3 [74]	12258	601	14.2	224	77.4	CVPR'24
Vim-Ti [81]	2117	1500	7.0	224	76.1	ICML'24
LocalVim-T [29]	1003	1500	8.0	224	76.2	arXiv'2403
EfficientVMamba-T [49]	6285	800	6.0	224	76.5	arXiv'2403
MobileMamba-S6	11000	652	15.0	**224**	**78.0**	CVPR'25
MobileMamba-S6†	11000	652	15.0	**224**	**80.7**	CVPR'25
MambaOut-Femto [71]	3350	1200	7.0	224	78.9	arXiv'2405
MobileOne-S4 [64]	3565	2978	14.8	224	79.4	CVPR'23
ConvNeXtV2-P [69]	4260	1370	9.1	224	79.7	CVPR'23
RepViT-M0.9 [67]	6427	832	5.1	224	78.7	CVPR'24
MPViT-T [31]	2121	1600	5.8	224	78.2	CVPR'22
EMO-6M [76]	4038	961	6.1	224	79.0	ICCV'23
SHViT-S4 [74]	7455	986	16.5	256	79.4	CVPR'24
ViL-T [1]	741	1500	6.0	224	78.3	arXiv'2406
PlainMamba-L1 [70]	1335	3000	7.0	224	77.9	BMVC'24
EfficientVMamba-S [49]	3327	1300	11.0	224	78.7	arXiv'2403
MobileMamba-B1	6986	1080	17.1	**256**	**79.9**	CVPR'25
MobileMamba-B1†	6986	1080	17.1	**256**	**82.2**	CVPR'25
FasterNet-S [3]	3731	4560	31.1	224	81.3	CVPR'23
ConvNeXtV2-N [69]	2987	2450	15.6	224	81.2	CVPR'23
MPViT-XS [31]	1663	2900	10.5	224	80.9	CVPR'22
EfficientViT-M4r384 [37]	3721	1486	12.4	384	79.8	CVPR'23
SHViT-S4r384 [74]	3549	2225	16.5	384	81.0	CVPR'24
ViL-S [1]	360	5100	23.0	224	81.5	arXiv'2406
VRWKV-S [11]	1598	4600	23.8	224	80.1	ICLR'25
LocalVim-S [29]	312	4800	28.0	224	81.2	arXiv'2403
MobileMamba-B2	3175	2427	17.1	**384**	**81.6**	CVPR'25
MobileMamba-B2†	3175	2427	17.1	**384**	**83.3**	CVPR'25
InceptionNeXt-T [73]	2335	4200	28.0	224	82.3	CVPR'24
MobileViTv3-1.0 [65]	1621	4220	5.1	384	79.7	arXiv'2209
EfficientViT-M5r512 [37]	1694	2670	12.4	512	80.8	CVPR'23
SHViT-S4r512 [74]	1993	3973	16.5	512	82.0	CVPR'24
ViL-B [1]	168	18600	89.0	224	82.4	arXiv'2406
VRWKV-B [11]	549	18200	93.7	224	82.0	ICLR'25
PlainMamba-L2 [70]	646	8100	25.0	224	81.6	BMVC'24
Vim-S [81]	932	5100	26.0	224	80.5	ICML'24
EfficientVMamba-B [49]	648	4000	33.0	224	81.8	arXiv'2403
MobileMamba-B4	1862	4313	17.1	**512**	**82.5**	CVPR'25
MobileMamba-B4†	1862	4313	17.1	**512**	**83.6**	CVPR'25

Table 3. **Classification Performance** on ImageNet-1K [10] dataset. White, gray, yellow, and blue backgrounds indicate *CNN-based, Transformer-based, Mamba/RWKV-based* and our MobileMamba, respectively. This kind of display continues for all subsequent experiments. † indicates the use of training strategies.

To demonstrate the scaling capability of the lightweight models, we maintain the network architecture of the MobileMamba-B1 model and increase the input resolution to 384^2 and 512^2, resulting in models with about 2G and 4G FLOPs, respectively. The MobileMamba-B2 and B4 models achieve higher classification results while having fewer FLOPs compared to other models. Additionally, the use of training strategies † further enhances model performance. For instance, applying training strategies to the MobileMamba-T2† model results in a +3.3↑ increase in Top-1 and a +1.7↑ increase in Top-5. Across all model scales, training strategies consistently demonstrate their ability to significantly improve performance.

Mask R-CNN Object Detection & Instance Segmentation on COCO							
Backbone	mAP^b	mAP^b_{50}	mAP^b_{75}	mAP^m	mAP^m_{50}	mAP^m_{75}	TP
EfficientNet-B0[59]	31.9	51.0	34.5	29.4	47.9	31.2	71
ResNet-50 [21]	38.0	58.6	41.4	34.4	55.1	36.7	41
FastViT-SA12 [63]	38.9	60.5	42.2	35.9	57.6	38.1	36
EfficientViT-M4[37]	32.8	54.4	34.5	31.0	51.2	32.2	121
PoolFormer-S12 [72]	37.3	59.0	40.1	34.6	55.8	36.9	32
EfficientFormer-L1 [32]	37.9	60.3	41.0	35.4	57.3	37.3	45
SHViT-S4 [74]	39.0	61.2	41.9	35.9	57.9	37.9	136
EMO-5M [76]	39.3	61.7	42.4	36.4	58.4	38.7	97
MobileMamba-B1	**40.6**	**61.8**	**43.8**	**37.4**	**58.9**	**39.9**	**152**
RetinaNet Object Detection on COCO							
Backbone	mAP^b	mAP^b_{50}	mAP^b_{75}	mAP^b_S	mAP^b_M	mAP^b_L	TP
MobileNetV3 [24]	29.9	49.3	30.8	14.9	33.3	41.1	153
EfficientViT-M4 [37]	32.7	52.2	34.1	17.6	35.3	46.0	160
PVTv2-B0 [68]	37.2	57.2	39.5	23.1	40.4	49.7	71
MobileFormer-508M [6]	38.0	58.3	40.3	22.9	41.2	49.7	58
EdgeViT-XXS [48]	38.7	59.0	41.0	22.4	42.0	51.6	60
SHViT-S4 [74]	38.8	59.8	41.1	22.0	42.4	52.7	**186**
EMO-5M [76]	38.9	59.8	41.0	**23.8**	42.2	51.7	138
EfficientVMamba-T [49]	37.5	57.8	39.6	22.6	40.7	49.1	42
MobileMamba-B1	**39.6**	**59.8**	**42.4**	21.5	**43.1**	**53.9**	181

Table 4. **Object Detection** and **Instance Segmentation** results by RetinaNet [34] and Mask RCNN [22] on MS-COCO 2017 [35].

Backbone	Reso.	FLOPs ↓	#Params ↓	mAP	TP↑
MobileNetv2 [53]	320	0.8G	4.3	22.1	5161
MobileNetv3 [23]	320	0.6G	5.0	22.0	5374
MobileViTv2-0.5 [47]	320	0.9G	2.0	21.2	4994
EMO-1M [76]	320	0.6G	2.3	22.0	2623
MobileMamba-B1	320	1.7G	18.0	**24.0**	3665
ResNet50 [21]	512	8.8G	26.6	25.2	692
MobileOne-S3 [64]	512	4.3G	13.1	27.3	652
MobileViTv2-0.75 [47]	512	1.8G	3.6	24.6	1231
EMO-2M [76]	512	0.9G	3.3	25.2	580
MobileViTv2-1.25 [47]	512	4.7G	8.2	27.8	632
EMO-5M [76]	512	1.8G	6.0	27.9	350
MobileMamba-B1	512	4.4G	18.0	**29.5**	1616

Table 5. **Object Detection** performance by SSDLite [23] on MS-COCO 2017 [35] dataset at 320×320 resolution.

4.3. MobileMamba on Downstream Tasks

Object Detection and Instance Segmentation. The pre-trained MobileMamba model is evaluated for object detection using light SSDLite [25] and heavy RetinaNet [34], as well as for instance segmentation with Mask R-CNN [22] on MS-COCO 2017 [35] dataset. For SSDLite in Tab. 5, we initially experiment at 320^2 resolution and subsequently increase to 512^2 while keeping other parameters constant. MobileMamba-B1 achieves +2↑ compared to EMO-1M [76] at 320^2 resolution. MobileMamba-B1 has -0.3G↓ FLOPs than MViTv2-1.25 [47] while achieving +1.7↑ in mAP at 512^2. For RetinaNet in Tab. 4, MobileMamba-B1 demonstrates a GPU throughput ×4.3↑ higher than EfficientVMamba-T [49], with +2.1↑ in mAP^b. Compared to EMO-5M [76], it shows +31%↑ in GPU throughput and +0.7↑ in mAP^b. For Mask R-CNN in Tab. 4, MobileMamba-B1 shows +57%↑ in throughput compared to EMO-5M [76], with +1.3↑ and +1.0↑ in mAP^b and mAP^m, respectively. Compared to SHViT-S4 [74], it achieves +1.6↑ and 1.5↑ in mAP^b and mAP^m.

Semantic Segmentation. The pre-trained MobileMamba is evaluated for semantic segmentation performance us-

	Backbone	TP↑	FLOPs↓	#Params.↓	mIoU
DeepLabv3 [5]	MobileOne [64]	518	14.7G	28.4	36.2
	MobileViTv2-0.5 [47]	1980	26.1G	6.3	31.9
	MobileViTv3-0.5 [66]	1737	-	6.3	33.5
	EMO-1M [76]	688	2.4G	5.6	33.5
	MobileViTv2-0.75 [47]	1160	40.0G	9.6	34.7
	MobileViTv3-0.75 [66]	1018	-	9.7	36.4
	EMO-2M [76]	470	3.5G	6.6	35.3
	MobileMamba-B4	1569	4.7G	23.0	**36.6**
Semantic FPN [30]	RepViT-M1.1 [67]	460	28.3G	12.5	40.6
	EMO-1M [76]	365	22.5G	5.2	34.2
	PVTv2-B0 [68]	382	25.0G	7.6	37.2
	EMO-2M [76]	293	23.5G	6.2	37.3
	EdgeViT-XXS [48]	310	24.4G	7.9	39.7
	EdgeViT-XS [48]	253	27.7G	10.6	41.4
	PVTv2-B1 [68]	287	34.2G	17.8	42.5
	EMO-5M [76]	214	25.8G	8.9	40.4
	MobileMamba-B4	1282	5.6G	19.8	**42.5**
PSPNet [78]	MobileNetv2 [53]	1926	53.1G	13.7	29.7
	MobileViTv2-0.5 [47]	1529	15.4G	3.6	31.8
	EMO-1M [76]	630	2.1G	4.3	33.2
	MobileViTv2-0.75 [47]	1014	26.6G	6.2	35.2
	MobileViTv2-1.0 [47]	826	40.3G	9.4	36.5
	EMO-2M [76]	442	3.1G	5.5	34.5
	MobileMamba-B4	1346	4.5G	20.5	**36.9**

Table 6. **Semantic Segmentation** results by DeepLabv3 [5], Semantic FPN [30], and PSPNet [78] on ADE20K [80] dataset at 512×512 resolution.

ing DeepLabv3 [5], Semantic FPN [30], and PSPNet [78] on the ADE20K [80] dataset in Tab. 6. For DeepLabv3, MobileMamba-B4 achieves +1.3↑ in mIoU over EMO-2M [76]. For Semantic FPN, MobileMamba demonstrates a significant advantage. Compared to EMO-5M [76], it has only 22% of the FLOPs while +2.1↑ in mIoU. Compared to EdgeViT-XS [48], it achieves +1.1↑ mIoU with only 20% of the FLOPs. Compared to PVTv2-B1 [68] with similar results, the FLOPs of our model are only 16% of theirs. For PSPNet, MobileMamba-B4 achieves +2.4↑ in mIoU over EMO-2M [76]. Compared to MobileViTv2-1.0 [47], it achieves +0.4↑ higher mIoU with only 11% of the FLOPs.

4.4. Extra Ablation and Explanatory Analysis

Incremental Experiments. Fig. 5 illustrates the process of deriving the MobileMamba model from the baseline EfficientViT-M5 [37] model through incremental experiments. Since FLOPs do not always fully reflect the inference speed of a model, we include the GPU throughput metric to demonstrate the model's efficiency. At the structural level, the Cascaded Group Attention is gradually transformed into the proposed MRFFI module. This process integrates multi-scale receptive fields while enhancing the model's throughput. Subsequently, a Fine-Grained design is applied to enhance the model's representation capabilities in terms of Mamba's global receptive field and frequency domain details, thereby improving model performance. *Reducing the number of network layers while increasing the dimension and global proportion ξ can decrease FLOPs and simultaneously increase throughput with similar accuracy.* Finally, *by decreasing d_state and scan directions and increasing the expanding ratio of SSM, the model's performance is further enhanced, and its throughput is significantly increased.* Ultimately, compared to the baseline, MobileMamba achieves an improvement of +0.9↑ in Top-1 and +0.6↑ in Top-5 accuracy, while also increasing throughput by +729↑ images per second.

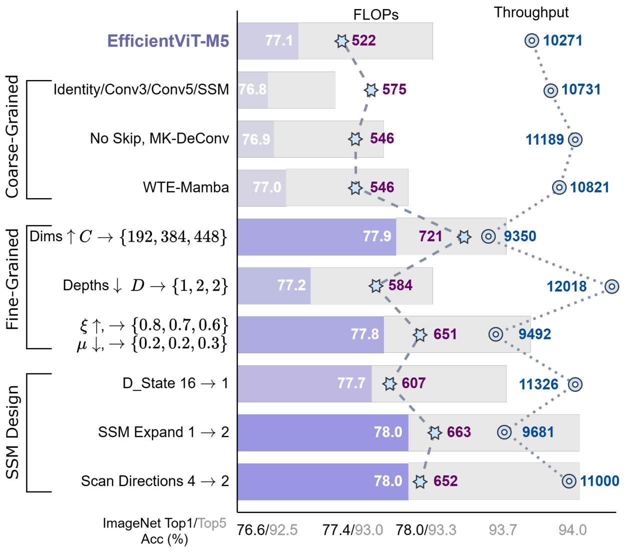


Figure 5. **Incremental Experiments** on the ImageNet-1K for MobileMamba compare Top1/Top5 Acc., FLOPs, and Throughput.

Efficiency Comparison. Tab. 7 presents a comparison with SoTA methods in efficiency and effectiveness. MobileMamba surpasses all methods in GPU throughput. On average, the three different sizes of MobileMamba models achieve ×3.5 ↑ faster in GPU throughput compared to EfficientVMamba [49]. However, for CPU throughput on AMD EPYC 9K84 96-Core and latency and latency on the mobile iPhone15 (ms), MobileMamba lags behind Transformer-based models. *This is attributed to the current engineering implementation of the Mamba method on CPUs, which still has room for improvement and optimization.* Nevertheless, compared to other Mamba-based methods, MobileMamba achieves only 15%-42% latency over EfficientVMamba on CPUs, while also achieving an average improvement of +1.5↑ in Top-1.

Ablations on Small Models with Low-Resolution. To enhance the performance of smaller models while increasing

Model	FLOPs (M)	#Params (M)	Reso.	Throughput		Latency		Top-1(%)
				GPU	CPU	CPU	Mobile	
EfficientViT-M2 [37]	201	4.2	224	18693	255	3.9	1.1	70.8
EMO-1M [76]	261	1.3	224	8361	91	10.9	5.1	71.5
MobileMamba-T2	255	8.8	192	21071	85	11.8	11.7	73.6
EfficientViT-M4 [37]	299	8.8	224	14612	228	4.4	1.6	74.3
EMO-2M [76]	439	2.3	224	6301	67	15.0	7.6	75.1
MobileMamba-T4	413	14.2	192	16571	84	11.8	16.9	76.1
EfficientViT-M5 [37]	522	12.4	224	10271	180	5.6	2.0	77.1
EfficientVMamba-T [49]	800	6.0	224	6285	14	70.0	113.6	76.5
MobileMamba-S6	652	15.0	224	11000	80	12.5	19.7	78.0
EfficientVMamba-S [49]	1300	11.0	224	3327	7	137.8	287.9	78.7
EMO-6M [76]	961	6.1	224	4038	42	23.5	14.8	79.0
MobileMamba-B1	1080	17.1	256	6986	49	20.3	47.0	79.9
EfficientViT-M5r512 [37]	2670	12.4	512	1694	31	32.4	3.5	80.8
EfficientVMamba-B [49]	4000	33.0	224	648	5	198.2	834.8	81.8
MobileMamba-B4	4313	17.1	512	1862	12	84.2	291.7	82.5

Table 7. Comparison with SoTAs in **Effectiveness** and **Efficiency**.

Reso.	FLOPs	Throughput	Params	Top-1
160	252	**21548**	11.7	72.6
192	255	21071	8.8	**73.5**
224	269	20203	6.5	73.3

(a) Small model for Lower Resolution.

Reso.	FLOPs	Throughput	Params	Top-1
224	2305	**3602**	50.7	80.0
384	2427	3175	17.1	**81.6**
224	4374	**2145**	79.4	80.7
512	4313	1862	17.1	**82.5**

(b) Large model for Higher Resolution.

Method	FLOPs	Throughput	Top-1
S6	652	9200	78.0
+KD	652	9200	80.0
+1000e	652	9200	80.7
+NLF	648	**11000**	**80.7**

(c) Training and Testing Strategies.

S	R	D	FLOPs	Throughput	Params	Top-1
4	1	1	607	**11782**	14.1	77.7
4	1	16	651	9945	14.3	77.8
4	2	1	663	10069	15.2	**78.1**
2	1	16	624	11121	14.1	77.6
2	2	1	652	11000	15.0	78.0

(d) Impact of Mamba Component.

n.	TP	Top-1
1	11000	78.0
2	10847	77.9
3	10791	78.0

(e) Impact of MK-DeConv.

Method	Params	FLOPs	Throughput	Top-1
wo WT	14.9	652	**11687**	77.8
w WT	15.0	652	11000	**78.0**

(f) Impact of Wave Transformation.

Table 8. **Ablation Studies** and comparison analysis on ImageNet-1K [10].

their throughput, we investigate the impact of input resolution. We set three input resolutions: 160^2, 192^2, and 224^2, and adjust the model parameters to ensure that the FLOPs are approximately *250M* for each resolution. As shown in Tab. 8a, *despite similar FLOPs, lower input resolutions result in higher model throughput and larger parameter sizes*. Considering throughput, parameter size, and performance, we design the small model with an input resolution of 192^2, achieving a good balance and satisfactory results.

Ablations on Large Models with High-Resolution. We explore ways to enhance the scaling capability of small models. In Tab. 8b, at the standard resolution of 224^2, increasing the model's depth and width to achieve *2G* and *4G* FLOPs does not significantly improve performance despite the increased computational load. *This is due to the excessively low input resolution in the current three-stage framework*. Therefore, we increase the input resolutions to 384^2 and 512^2. With similar FLOPs and a slight loss in throughput, the Top-1 improved by +1.6$\uparrow$ and +1.8$\uparrow$, respectively.

Effect of Training Strategies. Tab. 8c presents the incremental experiments using training and testing strategies. After applying KD, there is an increase of +2$\uparrow$ in Top-1 and +0.7$\uparrow$ in Top-5 accuracy on the ImageNet-1K dataset. Extending the training to 1000 epochs further improves these metrics by +0.7$\uparrow$ and +0.5$\uparrow$, respectively. Ultimately, the model with *652M* FLOPs achieves results of 80.7 in Top-1 and 95.2 in Top-5 accuracy, surpassing the model with *1080M* FLOPs that did not use the training strategies. Additionally, employing normalization fusion during the testing phase can further enhance the speed by $\times 1.2\% \uparrow$.

Ablations on Mamba Component. Experiments on the internal parameters of the Mamba model are shown in Tab. 8d. S, R, and D represent *scanning directions*, *expanding ratios* and *d_state*, respectively. Reducing the S can increase throughput, albeit with a slight decrease in performance. With the same number of S, using R=2 and D=1 results in higher throughput and better performance compared to R=1 and D=16. Therefore, the final choice is to use bidirectional scanning with an R=2 and D=1.

Ablations on MK-DeConv. We experimented with the number of splits n in the efficient MK-DeConv operation (see Tab. 8e). For $n = 1$, all channels use a single convolution module with a kernel size of 3. For $n = 3$, channels are split into three groups with $k = 3, 5, 7$ respectively and then concatenated along the channel dimension. The methods show no significant differences in parameters, FLOPs, and throughput yielding similar results. Thus, we adopt $n = 1$ for simplicity. However, using $k = 3$ results in an *ERF* of 3. After WT, the feature map size is halved, followed by convolution with the same $k = 3$ and an IWT, restoring the original feature size and effectively doubling the *ERF* to 6. *This approach achieves multi-kernel and multi-receptive field characteristics by combining single-branch convolutions and wavelet transformations.*

Effect of Wave Transformation Component. The wavelet transform generates one low-frequency and three high-frequency feature maps. The low-frequency map retains the original feature information, while the high-frequency maps capture edge details. Post wavelet transform, the halved feature maps undergo convolution and then inverse wavelet transform, restoring the original size and effectively doubling the receptive field. Despite a potential reduction in throughput, *the wavelet transform's benefits in enhancing the receptive field and extracting edge information can improve model performance* (see Tab. 8f).

5. Conclusion

We designed the MobileMamba framework to balance performance and efficiency, addressing the limitations of existing Mamba-based models. The proposed MRFFI module enhances perception across various receptive fields while preserving high-frequency features and inference efficiency. The training and testing strategies further enhance performance and efficiency. Extensive experiments on ImageNet-1K dataset validate the method's effectiveness, efficiency, and transferability in high-resolution downstream tasks.

Limitations and Future Work. Mamba models, despite their advancements, still exhibit engineering implementation shortcomings, including the need for substantial improvements in CPU acceleration and edge device acceleration. In the future, we will continue to concentrate on enhancing the inference capabilities of Mamba models across a range of devices, with a particular focus on efficiency.

Acknowledgments and Disclosure of Funding

This work was supported by Jianbing Lingyan Foundation of Zhejiang Province, P.R. China (Grant No. 2023C01022) and Major Project of Science and Technology of Yunnan Province, China under Grant 202402AD080001.

References

[1] Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-lstm: xlstm as generic vision backbone. *arXiv preprint arXiv:2406.04303*, 2024. 6

[2] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024. 5

[3] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: chasing higher flops for faster neural networks. In *CVPR*, 2023. 1, 3, 6

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[5] Liang-Chieh Chen. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 5, 7

[6] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 2022. 6

[7] MMSegmentation Contributors. Mmsegmentation: Openmmlab semantic segmentation toolbox and benchmark, 2020. 5

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 5

[9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR workshops*, 2020. 5

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 3, 5, 6, 8

[11] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv preprint arXiv:2403.02308*, 2024. 5, 6

[12] Shahaf E Finder, Roy Amoyal, Eran Treister, and Oren Freifeld. Wavelet convolutions for large receptive fields. In *ECCV*, 2025. 4

[13] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022. 2, 3

[14] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 3, 5

[15] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 2, 3

[16] 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.

[17] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *NeurIPS*, 2022. 3

[18] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *CVPR*, 2020. 2, 4

[19] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*, 2024. 3

[20] Haoyang He, Yuhu Bai, Jiangning Zhang, Qingdong He, Hongxu Chen, Zhenye Gan, Chengjie Wang, Xiangtai Li, Guanzhong Tian, and Lei Xie. Mambaad: Exploring state space models for multi-class unsupervised anomaly detection. In *NeurIPS*, 2024. 3

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2, 5, 6

[23] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 6

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 1, 2, 6

[25] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 2, 5, 6

[26] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[27] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*. Springer, 2016. 5

[28] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv preprint arXiv:2403.09338*, 2024. 2, 3

[29] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv preprint arXiv:2403.09338*, 2024. 2, 6

[30] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 2, 7

[31] Youngwan Lee, Jonghee Kim, Jeffrey Willette, and Sung Ju Hwang. Mpvit: Multi-path vision transformer for dense prediction. In *CVPR*, 2022. 1, 6

[32] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*, 2022. 1, 3, 6

[33] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *ICCV*, 2023. 1, 3, 6

[34] T Lin. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017. 2, 5, 6

[35] 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. 5, 6

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 5

[37] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *CVPR*, 2023. 1, 3, 6, 7

[38] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024. 2, 3

[39] Zhenhua Liu, Zhiwei Hao, Kai Han, Yehui Tang, and Yunhe Wang. Ghostnetv3: Exploring the training strategies for compact models. *arXiv preprint arXiv:2404.11202*, 2024. 2

[40] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[41] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[42] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024. 3

[43] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the ECCV (ECCV)*, 2018. 3, 6

[44] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *ECCV*. Springer, 2022. 1, 6

[45] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 1, 3

[46] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 2, 3

[47] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *TMLR*, 2023. 6, 7

[48] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *ECCV*, 2022. 3, 6, 7

[49] Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. *arXiv preprint arXiv:2403.09977*, 2024. 2, 3, 5, 6, 7

[50] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024. 5

[51] Tal Ridnik, Hussam Lawen, Asaf Noy, Emanuel Ben Baruch, Gilad Sharir, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture. In *WACV*, 2021. 5

[52] Jiacheng Ruan and Suncheng Xiang. Vm-unet: Vision mamba unet for medical image segmentation. *arXiv preprint arXiv:2402.02491*, 2024. 3

[53] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 6, 7

[54] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2

[55] Yuheng Shi, Minjing Dong, and Chang Xu. Multi-scale vmamba: Hierarchy in hierarchy visual state space model. *arXiv preprint arXiv:2405.14174*, 2024. 2

[56] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022. 2, 3

[57] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 5

[58] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[59] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1, 3, 6

[60] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *ICML*. PMLR, 2021. 3

[61] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Chao Xu, and Yunhe Wang. Ghostnetv2: Enhance cheap operation with long-range attention. In *NeurIPS*, 2022. 1, 2

[62] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 5

[63] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *ICCV*, 2023. 3, 6

[64] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *CVPR*, 2023. 6, 7

[65] Shakti N Wadekar and Abhishek Chaurasia. Mobilevitv3: Mobile-friendly vision transformer with simple and effective fusion of local, global and input features. *arXiv preprint arXiv:2209.15159*, 2022. 1, 3, 6

[66] Shakti N Wadekar and Abhishek Chaurasia. Mobilevitv3: Mobile-friendly vision transformer with simple and effective fusion of local, global and input features. *arXiv preprint arXiv:2209.15159*, 2022. 7

[67] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *CVPR*, 2024. 6, 7

[68] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *CVM*, 2022. 6, 7

[69] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *CVPR*, 2023. 6

[70] Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *arXiv preprint arXiv:2403.17695*, 2024. 3, 6

[71] Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? *arXiv preprint arXiv:2405.07992*, 2024. 6

[72] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *CVPR*, 2022. 1, 3, 6

[73] Weihao Yu, Pan Zhou, Shuicheng Yan, and Xinchao Wang. Inceptionnext: When inception meets convnext. In *CVPR*, pages 5672–5683, 2024. 6

[74] Seokju Yun and Youngmin Ro. Shvit: Single-head vision transformer with memory efficient macro design. In *CVPR*, 2024. 1, 3, 5, 6

[75] Hongyi Zhang. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5

[76] Jiangning Zhang, Xiangtai Li, Jian Li, Liang Liu, Zhucun Xue, Boshen Zhang, Zhengkai Jiang, Tianxin Huang, Yabiao Wang, and Chengjie Wang. Rethinking mobile block for efficient attention-based models. In *ICCV*, 2023. 1, 2, 3, 6, 7

[77] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 1, 3

[78] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 5, 7

[79] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. 5

[80] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019. 5, 7

[81] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 2, 3, 6