

MobileMamba: Lightweight Multi-Receptive Visual Mamba Network

Supplementary Material

Haoyang He^{1*} Jiangning Zhang^{1,2*} Yuxuan Cai³ Hongxu Chen¹ Xiaobin Hu²
 Zhenye Gan² Yabiao Wang² Chengjie Wang² Yunsheng Wu² Lei Xie^{1†}

¹Zhejiang University ²Youtu Lab, Tencent ³Huazhong University of Science and Technology

Code: <https://github.com/lewandofskye/MobileMamba>

Supplementary Material Overview

The supplementary material presents more comprehensive analysis and results of our MobileMamba to facilitate the comparison of subsequent methods:

- **Sec. A.1** provides more detailed Fine-Grained design analysis and experiments on ImageNet-1K [2] dataset.
- **Sec. A.2** provides more detailed Kernel Size analysis and experiments on ImageNet-1K [2] dataset.
- **Sec. A.3** provides more detailed DropPath analysis and experiments on ImageNet-1K [2] dataset.
- **Sec. A.4** provides more detailed ERF Visualization analysis compared with different structure SoTA methods on ImageNet-1K [2] dataset.
- **Sec. A.5** provides more detailed Pre-trained Models with Different Resolutions for Downstream Tasks analysis and experiments on MS-COCO 2017 [10] and ADE20K [16] dataset.
- **Sec. B.1** provides more detailed object detection results using different frameworks on MS-COCO 2017 [10] dataset.
- **Sec. B.2** provides more detailed semantic segmentation results using Mask R-CNN [3] for multiple magnitudes of MobileMamba on ADE20K [16] dataset.
- The **Codes** folder in the supplementary materials contains all the training and testing code for the models, as well as the log files for each model.

A. More Ablation and Explanatory Analysis

A.1. Fine-Grained design analysis

We conducted experiments to analyze the impact of global and local channel ratios in Tab. A1, dimensions in Tab. A2, and depth in Tab. A3. For the global and local channel ratios, we observed the importance of global channels for model performance, despite a slight decrease in throughput. In higher stages, due to the increased number of channels,

some redundancy may exist. To reduce computational load, we directly map 10% of the channels in the last two stages.

Regarding dimensionality, we controlled variables by maintaining similar FLOPs and throughput while adjusting the global and local ratios to accommodate different dimensional changes. Altering the dimensions in stage 1 significantly affects FLOPs and throughput, whereas changes in stage 3 primarily impact the number of model parameters. To maximize dimensions in each stage while maintaining low FLOPs and high throughput, we selected {192, 384, 448} as the dimensions for each stage.

For model depth, we found that increasing depth significantly reduces throughput. Therefore, we increased depth while maintaining similar throughput, but the effect was limited due to lower FLOPs under the same conditions. In extreme cases, where each stage has only one layer and FLOPs are balanced with other models, throughput is significantly higher, but performance is poor. After trade-offs, we chose a depth of {1, 2, 2}.

Table A1. Ablations on Global ξ and Local μ Ratios.

$\{C_1, C_2, C_3\}$ $\{D_1, D_2, D_3\}$	$\{\xi_1, \xi_2, \xi_3\}$ $\{\mu_1, \mu_2, \mu_3\}$	FLOPs (M)	Params (M)	Throughput	Top-1
{192, 384, 448}	{0.6, 0.6, 0.6}	620	14.6	11353	77.5
{1, 2, 2}	{0.4, 0.3, 0.3}				
{192, 384, 448}	{0.7, 0.6, 0.5}	619	14.3	11815	77.7
{1, 2, 2}	{0.2, 0.2, 0.3}				
{192, 384, 448}	{0.8, 0.6, 0.6}	637	14.7	11222	77.7
{1, 2, 2}	{0.2, 0.3, 0.3}				
{192, 384, 448}	{0.8, 0.7, 0.6}	652	15.0	10949	77.8
{1, 2, 2}	{0.2, 0.3, 0.4}				
{192, 384, 448}	{0.8, 0.8, 0.8}	675	16.0	10560	78.0
{1, 2, 2}	{0.2, 0.1, 0.1}				
{192, 384, 448}	{0.0, 0.7, 0.8}	618	14.8	12735	77.2
{1, 2, 2}	{1.0, 0.2, 0.1}				
{192, 384, 448}	{0.6, 0.7, 0.8}	646	15.6	11546	77.8
{1, 2, 2}	{0.4, 0.2, 0.1}				
{192, 384, 448}	{0.8, 0.7, 0.6}	652	15.0	11000	78.0
{1, 2, 2}	{0.2, 0.2, 0.3}				

*Equal contributions.

†Corresponding author.

Table A2. Ablations on Dimensions C .

$\{C_1, C_2, C_3\}$ $\{D_1, D_2, D_3\}$	$\{\xi_1, \xi_2, \xi_3\}$ $\{\mu_1, \mu_2, \mu_3\}$	FLOPs (M)	Params (M)	Throughput	Top-1
$\{192, 320, 368\}$ $\{1, 3, 4\}$	$\{0., 0.75, 0.75\}$ $\{0.9, 0.15, 0.15\}$	632	15.9	10811	77.7
$\{172, 320, 368\}$ $\{1, 3, 4\}$	$\{0.65, 0.65, 0.65\}$ $\{0.25, 0.25, 0.25\}$	581	15.1	10809	77.6
$\{180, 336, 368\}$ $\{1, 3, 4\}$	$\{0.5, 0.5, 0.5\}$ $\{0.4, 0.4, 0.4\}$	595	14.8	10916	77.6
$\{192, 336, 368\}$ $\{1, 3, 4\}$	$\{0.4, 0.4, 0.4\}$ $\{0.5, 0.5, 0.5\}$	599	14.2	11373	77.5
$\{208, 400, 464\}$ $\{1, 2, 2\}$	$\{0.6, 0.5, 0.4\}$ $\{0.3, 0.4, 0.5\}$	678	15.2	11120	77.8
$\{224, 336, 400\}$ $\{1, 2, 2\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	659	12.6	11337	77.4
$\{208, 384, 416\}$ $\{1, 2, 2\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	681	14.5	10956	78.1
$\{176, 384, 480\}$ $\{1, 2, 2\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	622	15.7	11599	77.8
$\{192, 384, 448\}$ $\{1, 2, 2\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	652	15.0	11000	78.0

Table A3. Ablations on Depth D .

$\{C_1, C_2, C_3\}$ $\{D_1, D_2, D_3\}$	$\{\xi_1, \xi_2, \xi_3\}$ $\{\mu_1, \mu_2, \mu_3\}$	FLOPs (M)	Params (M)	Throughput	Top-1
$\{160, 304, 448\}$ $\{2, 3, 3\}$	$\{0.15, 0.55, 0.55\}$ $\{0.35, 0.35, 0.35\}$	560	14.8	11236	77.6
$\{128, 256, 384\}$ $\{3, 4, 5\}$	$\{0.15, 0.55, 0.55\}$ $\{0.35, 0.35, 0.35\}$	510	15.0	11155	77.1
$\{192, 384, 448\}$ $\{1, 2, 2\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	652	15.0	11006	78.0
$\{208, 416, 624\}$ $\{1, 1, 1\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	648	15.3	12397	77.4
$\{192, 384, 576\}$ $\{1, 2, 1\}$	$\{0.8, 0.7, 0.6\}$ $\{0.2, 0.2, 0.3\}$	651	15.1	11000	77.8

A.2. Effect of kernel sizes

We experimented with the impact of different convolution kernel sizes across stages, as shown in Tab. A4. Using the same kernel size across different stages yields similar results. However, reducing the kernel size as the feature map scale decreases with increasing stages improves model performance.

Table A4. Ablations on Kernel Size

Size	FLOPs(M)	Params(M)	Throughput	Top-1
$\{7, 7, 7\}$	15.2	653	10937	77.7
$\{5, 5, 3\}$	15.0	652	11142	77.8
$\{5, 3, 3\}$	15.0	652	11130	77.6
$\{7, 5, 3\}$	15.0	652	11000	78.0

A.3. Effect of DropPath

For the MobileMamba-T2, T4, and S6 models, we did not use DropPath due to their shallow depth. In the B1 model, we applied DropPath, with specific results shown in Tab. A5. A DropPath value of 0.03 achieved the best performance, increasing Top-1 accuracy by 0.2 compared to not using DropPath. Further increasing the DropPath value did

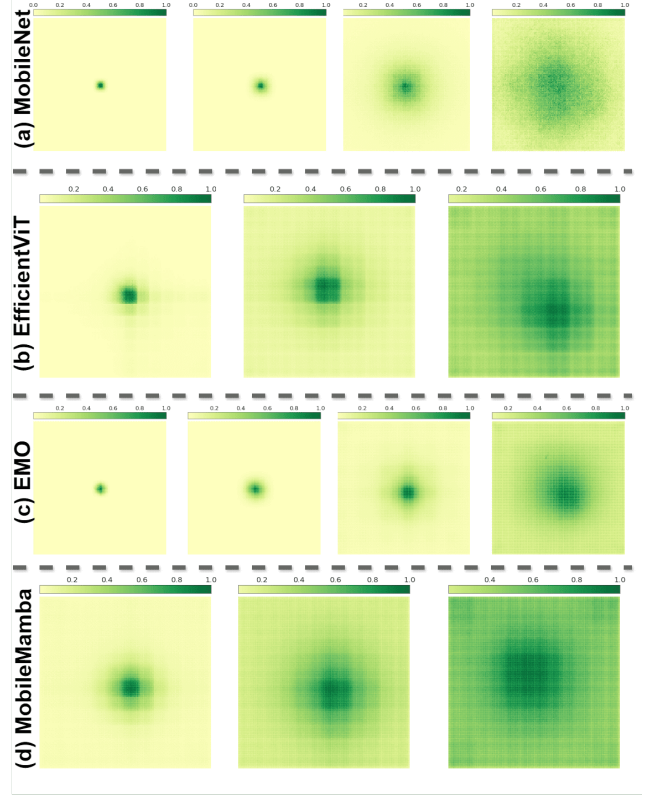


Figure A1. Visualization of the ERF of different model methods. not lead to additional performance improvements.

Table A5. Ablations on Drop-path rate.

Drop-path Rate	Top-1
0.0	79.7
0.03	79.9
0.05	79.8
0.07	79.8
0.1	79.8

A.4. Visualization of the ERF of different model methods

In Fig. A1, we compare the ERF visualization results of CNN-based MobileNet [5, 7, 12], Transformer-based EfficientViT [11], hybrid-structured EMO [14], and our MobileMamba at different stages. The input resolution is fixed at 224x224. Both our method and EfficientViT [11] employ a three-stage approach, while MobileNet [5, 7, 12] and EMO [14] follow the traditional four-stage approach. Our MobileMamba method exhibits a larger and more intense ERF at each stage compared to the other SoTAs.

A.5. Analysis of Pre-trained Models with Different Resolutions for Downstream Tasks

The specific experimental results for downstream tasks are shown in Tab. A6, A7, A8. We investigate the impact

of pre-trained model weights with different input resolutions on downstream tasks. We use two pre-trained model weights, MobileMamba-B1 and MobileMamba-B4. The only difference between them is the resolution used during pre-training on ImageNet-1K [2]: MobileMamba-B1 is pre-trained at a resolution of 256, while MobileMamba-B4 is pre-trained at a resolution of 512. All other model parameters are identical. For the object detection task in downstream tasks, MobileMamba-B1 outperforms MobileMamba-B4 on all metrics in SSDLite [6], RetinaNet [9], and Mask RCNN [3]. Conversely, for the semantic segmentation task, MobileMamba-B4 outperforms MobileMamba-B1 on all metrics in DeepLabv3 [1], Semantic FPN [8], and PSPNet [15]. This may be because object detection tasks require stronger semantic feature information, while semantic segmentation tasks demand higher segmentation accuracy. MobileMamba-B4, pre-trained at a high resolution of 512, extracts features with higher segmentation accuracy but slightly weaker semantic information. In contrast, MobileMamba-B1, pre-trained at a lower resolution of 256, extracts features with stronger semantic information but lower accuracy. Therefore, we use MobileMamba-B1 pre-trained weights as the backbone for object detection tasks to enhance semantic information extraction. For semantic segmentation tasks, we use MobileMamba-B4 pre-trained weights as the backbone to improve segmentation accuracy.

Table A6. Detailed object detection performance using SSDLite [4] and RetinaNet [9] of our MobileMamba on MS-COCO 2017 [10] dataset. †: 512×512 resolution.

	Backbone	#Params ↓	FLOPs ↓	mAP	mAP_{50}^b	mAP_{75}^b	mAP_S^b	mAP_M^b	mAP_L^b
SSDLite [4]	MobileMamba-B1	18.0	1.7G	24.0	39.5	24.0	3.1	23.4	46.9
	MobileMamba-B4	18.0	1.7G	23.9	39.5	24.2	2.9	23.5	47.1
	MobileMamba-B1†	18.0	4.4G	29.5	47.7	30.4	8.9	35.0	47.0
	MobileMamba-B4†	18.0	4.4G	29.1	47.1	30.0	8.7	34.3	46.7
RetinaNet [9]	MobileMamba-B1	27.1	151G	39.6	59.8	42.4	21.5	43.1	53.9
	MobileMamba-B4	27.1	151G	39.5	59.9	42.1	21.5	42.9	54.6

B. Detailed Downstream Results

B.1. Detailed Object Detection Results

Tab. A6 shows more detailed object detection results using SSDLite [4] and RetinaNet [9] of our MobileMamba on MS-COCO 2017 [10] dataset, while Tab. A7 provide detailed object detection results using Mask R-CNN [3].

Table A7. Detailed object detection performance using Mask RCNN [3] of our MobileMamba on MS-COCO 2017 [10] dataset.

Backbone	#Params ↓	FLOPs ↓	mAP	mAP_{50}^b	mAP_{75}^b	mAP_S^b	mAP_M^b	mAP_L^b
			mAP	mAP_{50}^m	mAP_{75}^m	mAP_S^m	mAP_M^m	mAP_L^m
MobileMamba-B1	38.0	178G	40.6	61.8	43.8	22.4	43.5	55.9
			37.4	58.9	39.9	17.1	39.9	56.4
MobileMamba-B4	38.0	178G	40.1	61.8	43.0	22.0	42.9	56.1
			36.9	58.6	39.2	16.4	39.0	56.8

Table A8. Detailed semantic segmentation performance using DeepLabv3 [1], Semantic FPN [8], and PSPNet [15] to adequately evaluate our MobileMamba on ADE20K [16] dataset.

	Backbone	#Params ↓	FLOPs ↓	mIoU	aAcc	mAcc
DeepLabv3 [1]	MobileMamba-B1	23.0	4.7G	36.7	76.0	46.8
	MobileMamba-B4	23.0	4.7G	36.6	76.3	47.1
FPN [8]	MobileMamba-B1	19.8	5.6G	40.7	79.4	51.8
	MobileMamba-B4	19.8	5.6G	42.5	79.9	53.7
PSPNet [15]	MobileMamba-B1	20.5	4.5G	36.5	76.2	46.7
	MobileMamba-B4	20.5	4.5G	36.9	76.2	47.9

B.2. Detailed Semantic Segmentation Results

Tab. A8 shows more detailed semantic segmentation results using DeepLabv3 [1], Semantic FPN [8], SegFormer [13], and PSPNet [15] of our MobileMamba on ADE20K [16] dataset.

References

- [1] Liang-Chieh Chen. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3
- [2] 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. 1, 3
- [3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 3
- [4] 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. [3](#)
- [5] 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](#)
- [6] 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. [3](#)
- [7] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. [2](#)
- [8] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. [3](#)
- [9] T Lin. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017. [3](#)
- [10] 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. [1](#), [3](#)
- [11] 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. [2](#)
- [12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. [2](#)
- [13] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. [3](#)
- [14] 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. [2](#)
- [15] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. [3](#)
- [16] 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. [1](#), [3](#)