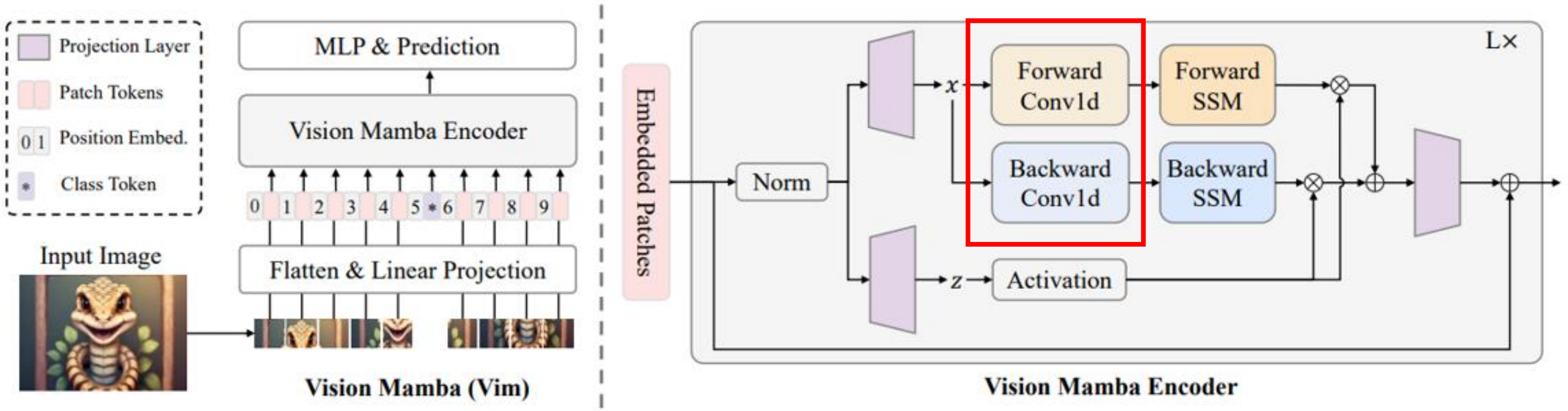
QuadMamba: Learning Quadtree-based Selective Scan for Visual State Space Model

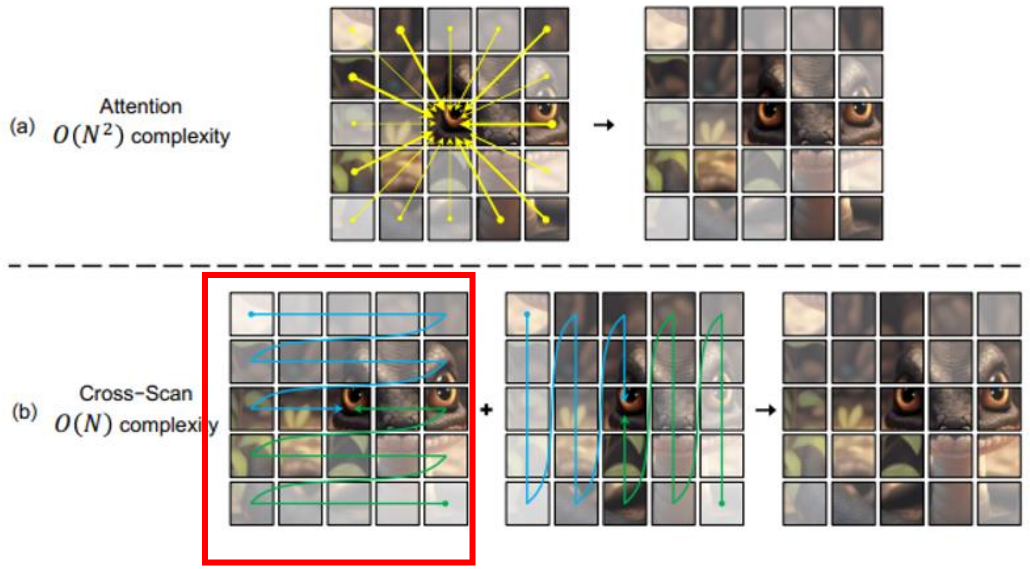
Fei Xie[1], **Weijia Zhang**[1], **Zhongdao Wang**[2], **Chao Ma**[1]*
[1]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2]Huawei Noah's Ark Lab
{jaffe031, weijia.zhang, chaoma}@sjtu.edu.cn
wangzhongdao.edu@huawei.com

Background



Vision Mamba-ICML24

Bi-directional Scanning

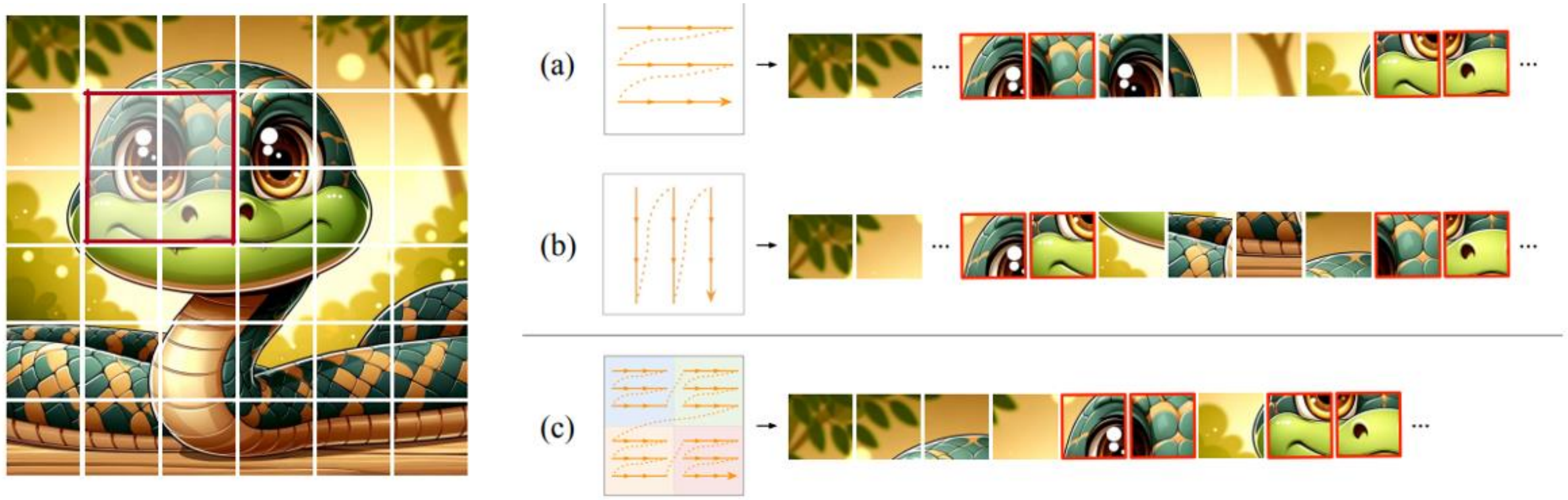


Vmamba-NeurIPS24

Cross Scanning

Background

LocalMamba-WACV24



Weakness:

data bias,
not flexible,
handcrafted-design, scanning.

Background

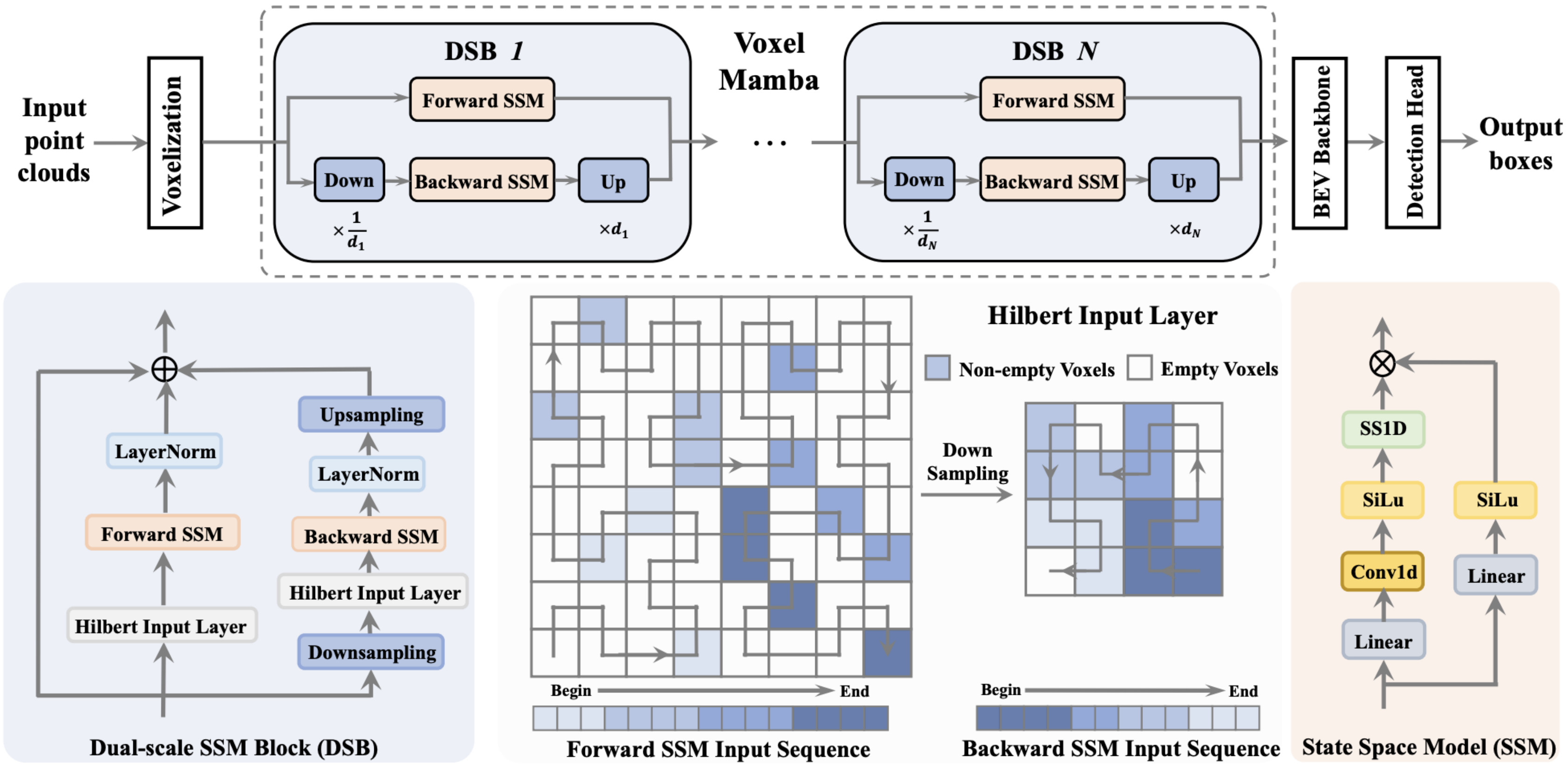
VoxelMamba, arXiv 2024

1) **Hilbert scan on voxels**

2) **Dual-Scale SSM**
   2 directions & 2 scales

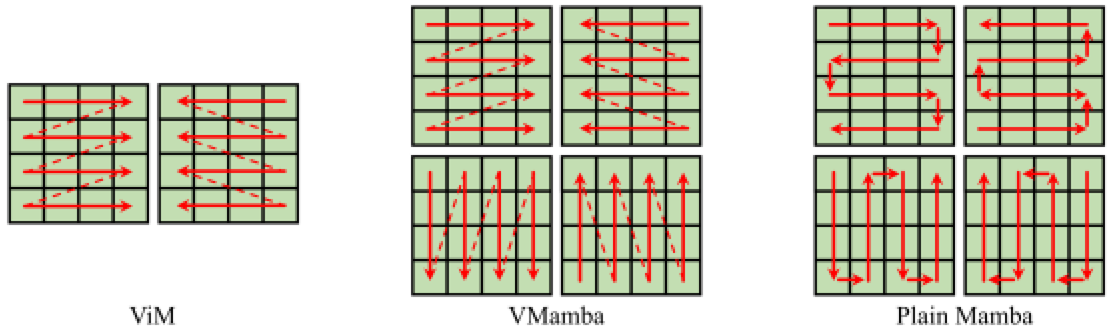
3) **Implicit Window Partition**
   a) Preserves 3D positional info in voxel features
   b) Enhances spatial locality
   c) Done by encoding voxel token 3D coords via MLP (resemble PCM)

$$\mathbf{MLP}(\text{concat}(z, \lfloor\frac{x^i}{w}\rfloor, \lfloor\frac{y^i}{h}\rfloor, x^i \bmod w, y^i \bmod h)), i = 0, 1$$



PlainMamba, arXiv 2024

PlainMamba (classification, segmentation, 2OD, instance segmentation)

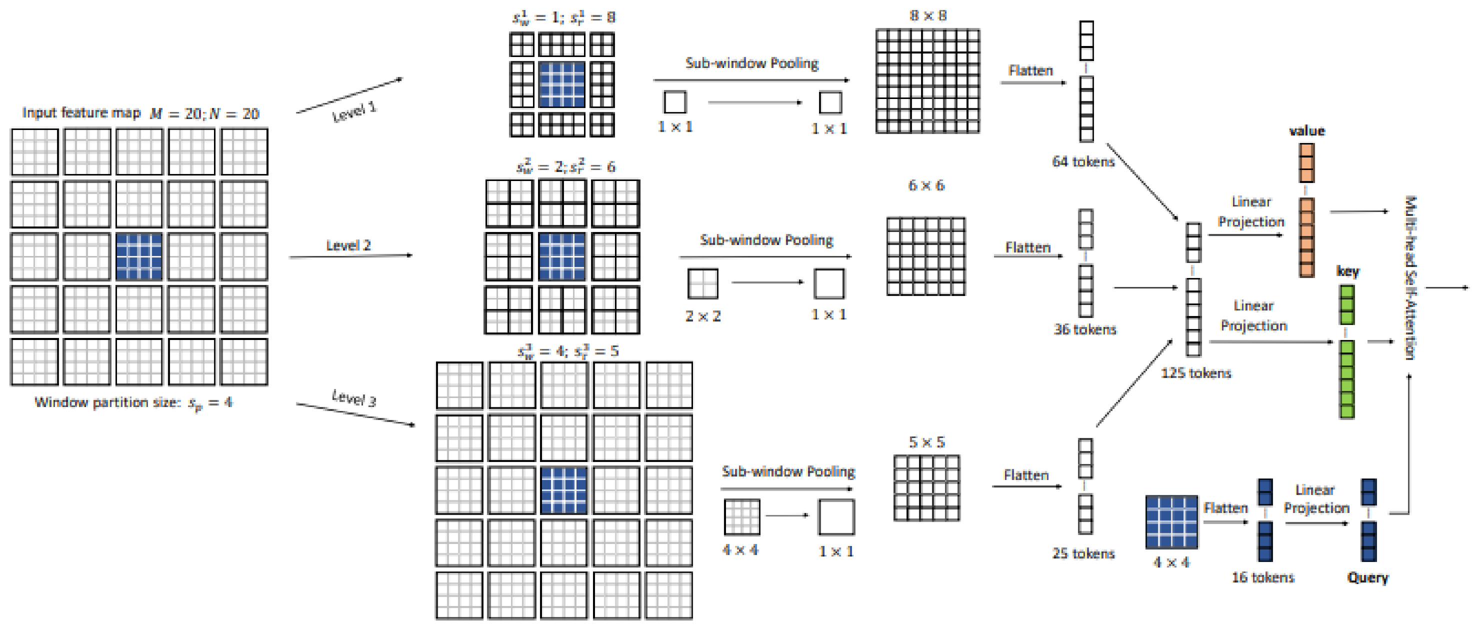


$$h'_{k,i} = \bar{\mathbf{A}}_i h_{k,i-1} + (\bar{\mathbf{B}}_i + \bar{\mathbf{\Theta}}_{k,i}) x_i$$

$$y'_i = \sum_{k=1}^{4} (\mathbf{C}_i h'_{k,i} + \mathbf{D} x_i), \quad y_i = y'_i \odot z_i$$

1) continuous 2D scan 2) direction-aware update

Background



Focal Transformer in NeurIPS2023

Adaptive region in Vision Transformer

Motivation

High-level Mid-level Finest-level

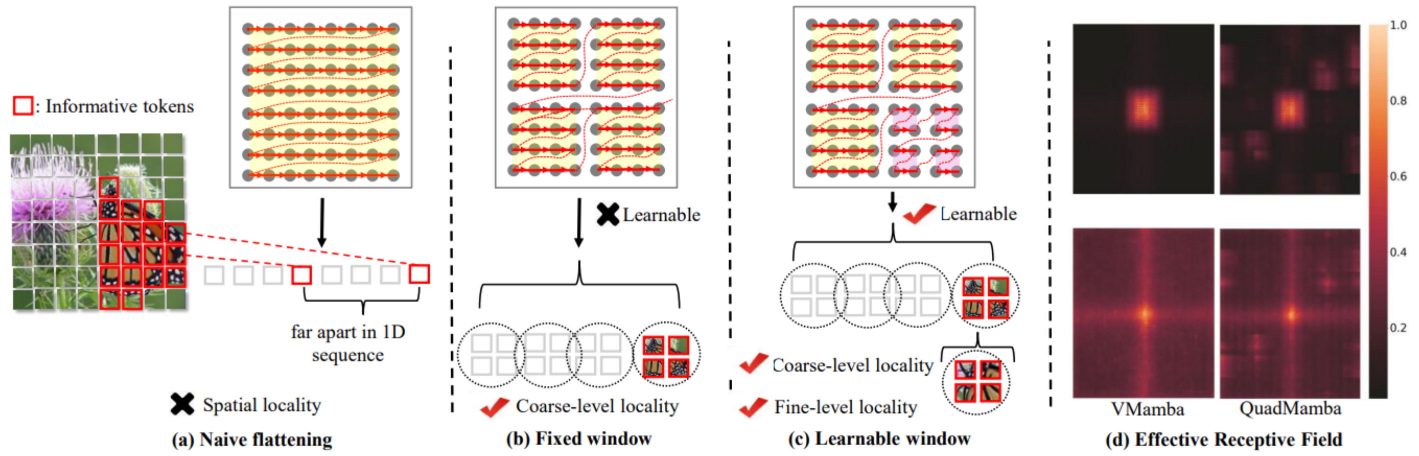
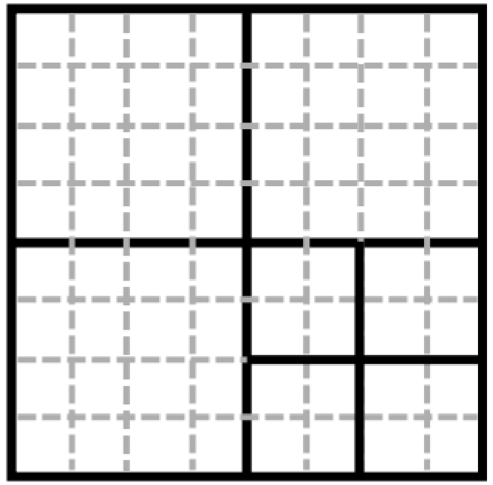
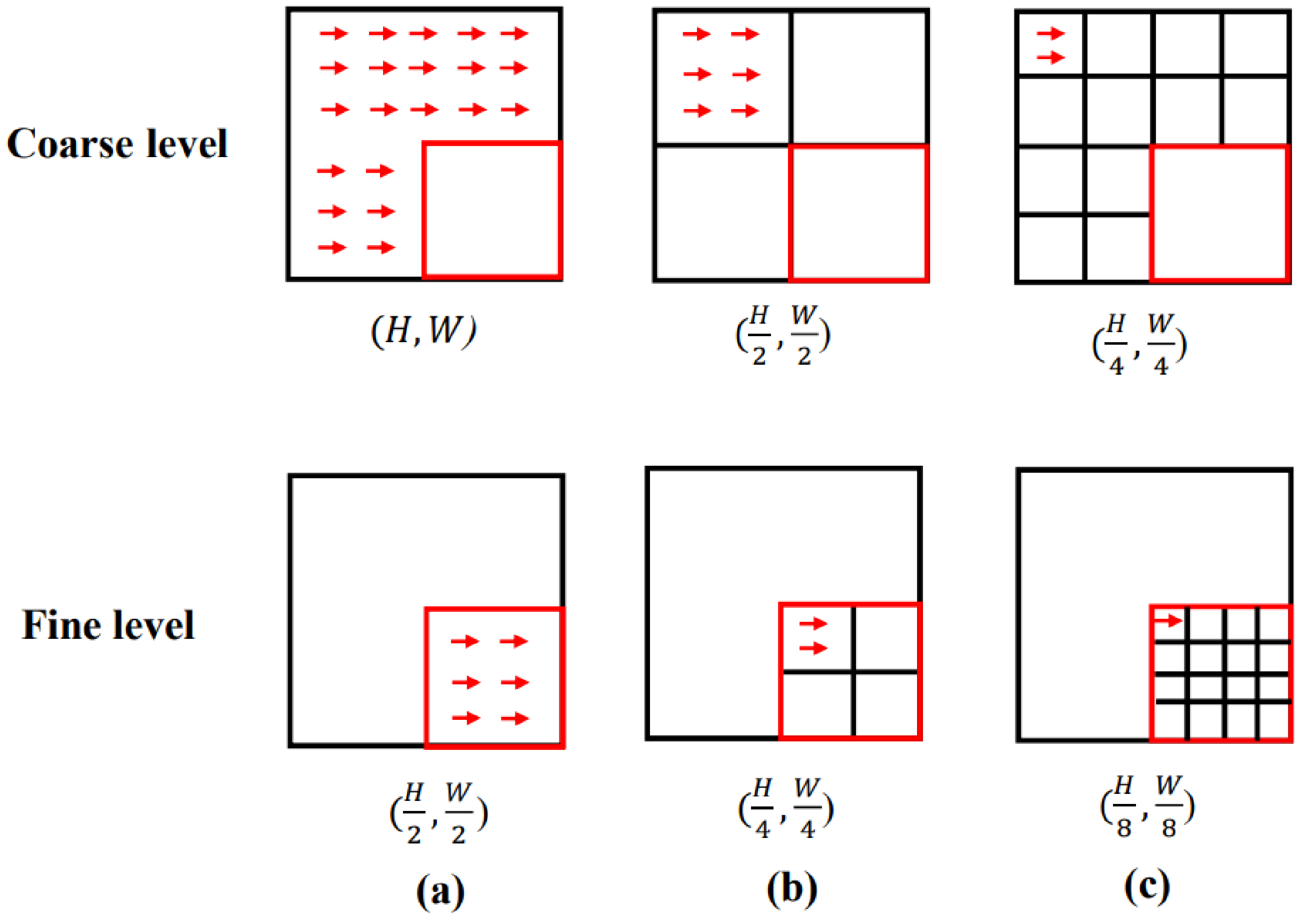


Figure 1: Illustration of scan strategies for transforming 2D visual data into 1D sequences. (a) naive raster scan [80, 41, 66] ignores the 2D locality; (b) fixed window scan [26] lacks the flexibility to handle visual signals of varying granularities; (c) our learnable window partition and scan strategy adaptively preserves the 2D locality with a focus on the more informative window quadrant; (d) the effective receptive field of our QuadMamba demonstrates more locality than the plain Vision Mamba.



Motivation



Coarse level
Fine level
(H,W)
$(\frac{H}{2},\frac{W}{2})$
$(\frac{H}{4},\frac{W}{4})$
$(\frac{H}{2},\frac{W}{2})$
$(\frac{H}{4},\frac{W}{4})$
$(\frac{H}{8},\frac{W}{8})$
(a)
(b)
(c)

Method

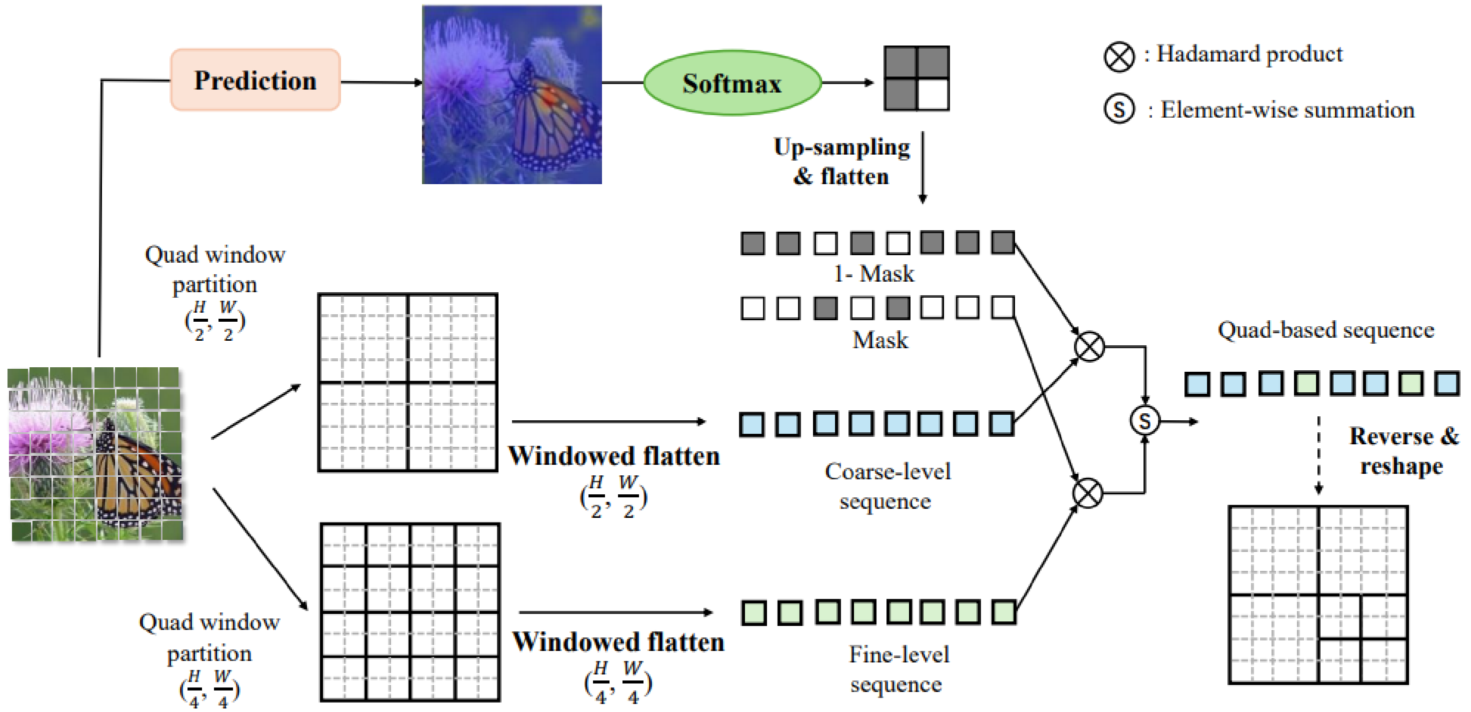

Prediction
Softmax
Up-sampling & flatten
⊗ : Hadamard product
Ⓢ : Element-wise summation
Quad window partition $(\frac{H}{2}, \frac{W}{2})$
1- Mask
Mask
Quad-based sequence
Windowed flatten $(\frac{H}{2}, \frac{W}{2})$
Coarse-level sequence
Reverse & reshape
Quad window partition $(\frac{H}{4}, \frac{W}{4})$
Windowed flatten $(\frac{H}{4}, \frac{W}{4})$
Fine-level sequence

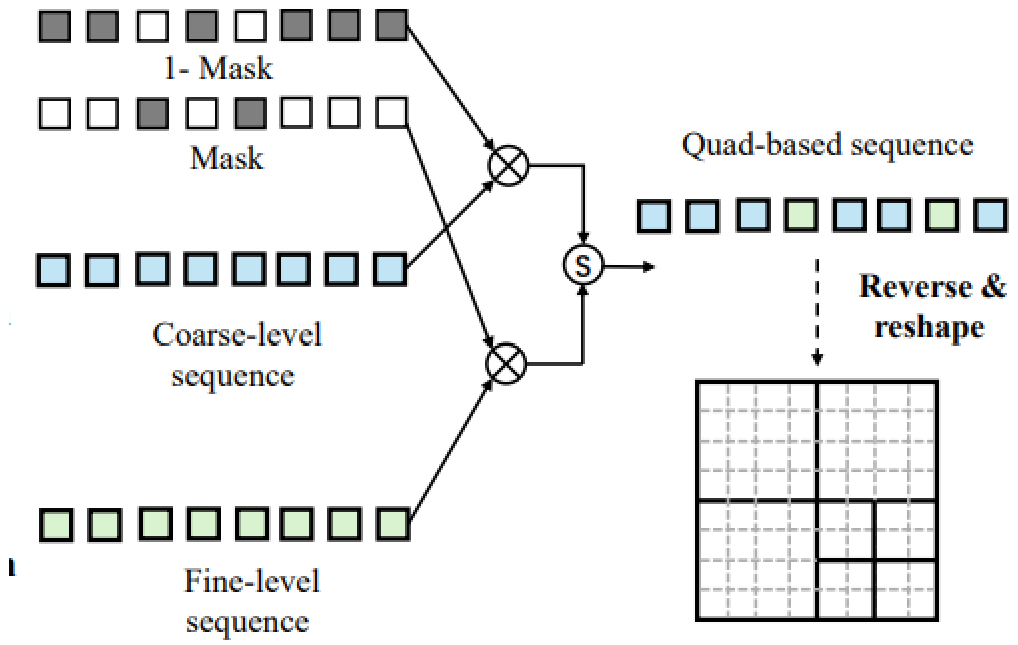
Figure 3: Quadtree-based selective scan with prediction modules. Image tokens are partitioned into bi-level window quadrants from coarse to fine. A fully differentiable partition mask is then applied to generate the 1D sequence with negligible computational overhead.

Method

Partition map prediction. The image feature $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$, containing a total of $N = HW$ embedding tokens, is first projected into score embeddings $\boldsymbol{x}_{\mathrm{s}}$:

$$\boldsymbol{x}_{\mathrm{s}} = \phi_{\mathrm{s}}(\boldsymbol{x}), \quad \boldsymbol{x}_{\mathrm{s}} \in \mathbb{R}^{N\times C}, \tag{4}$$

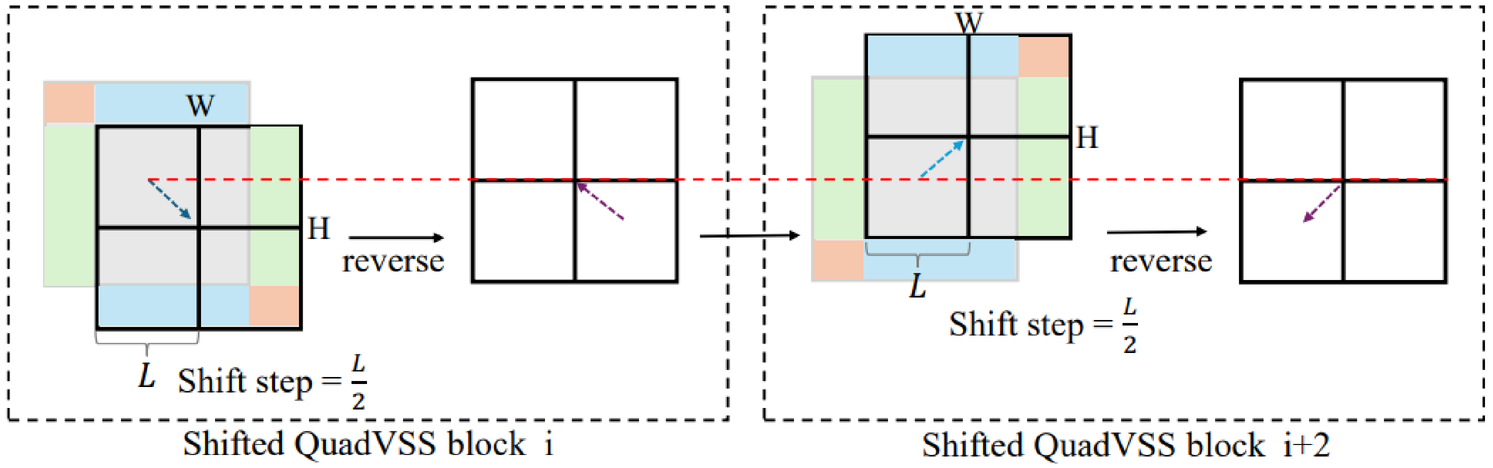
Method

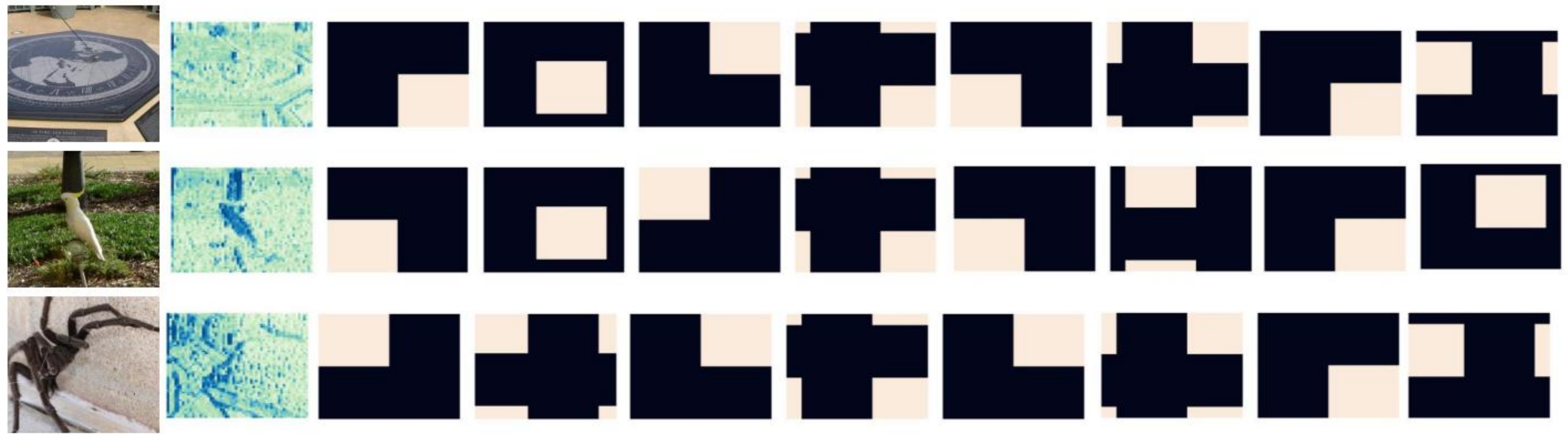


Differentiable 1D sequence construction. Although our target is to conduct adaptive learned window quadrant partition, it is non-trivial to construct the 1D token sequence from the 2D spatial windowed features. Directly sampling from the 2D feature according to the learned index for each sequence token is non-differentiable, which impedes end-to-end training. To overcome this, we apply a sequence masking strategy to formulate the token sequence, implemented by the Gumbel-Softmax technique

Method

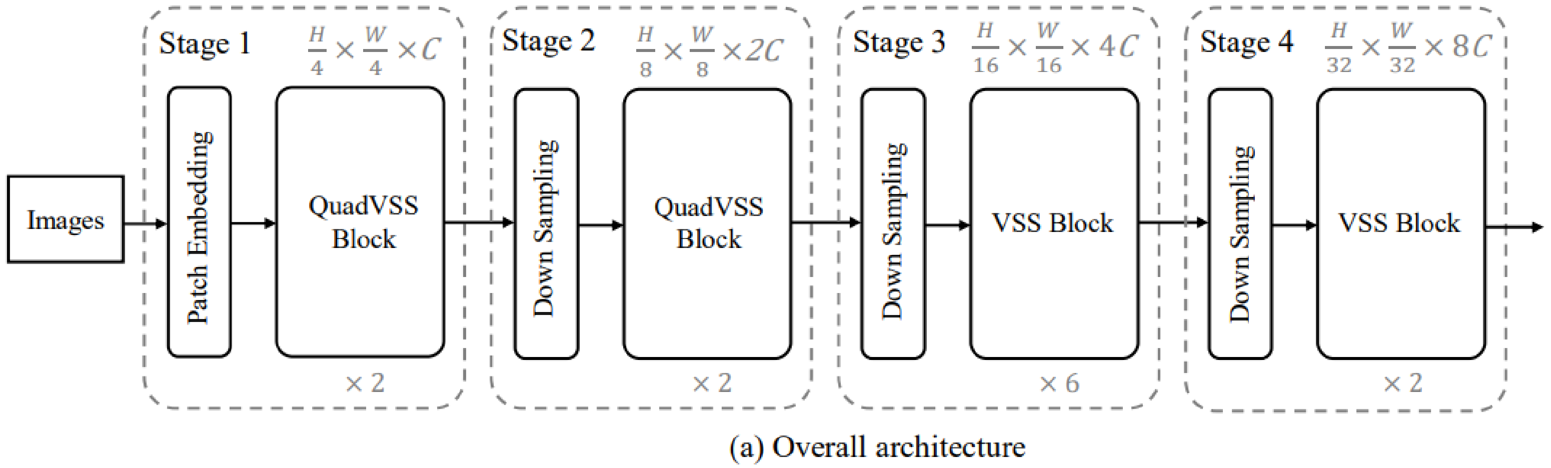
Omnidirectional window shifting scheme


W
H
reverse
L
Shift step = $\frac{L}{2}$
Shifted QuadVSS block i
W
H
reverse
L
Shift step = $\frac{L}{2}$
Shifted QuadVSS block i+2



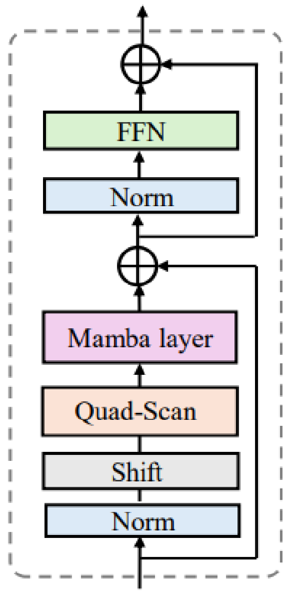
全向窗口移动机制确保所有区域进行完整的重要性预测

Method



Images
Stage 1 $\frac{H}{4} \times \frac{W}{4} \times C$
Patch Embedding
QuadVSS Block
× 2
Stage 2 $\frac{H}{8} \times \frac{W}{8} \times 2C$
Down Sampling
QuadVSS Block
× 2
Stage 3 $\frac{H}{16} \times \frac{W}{16} \times 4C$
Down Sampling
VSS Block
× 6
Stage 4 $\frac{H}{32} \times \frac{W}{32} \times 8C$
Down Sampling
VSS Block
× 2

(a) Overall architecture



(b) QuadVSS Block

- Lite – block:$\{2, 2, 2, 2\}$, QuadVSS stages:$\{1, 2\}$, #Params: 5.4M, FLOPs: 0.82G.
- Tiny – block:$\{2, 6, 2, 2\}$, QuadVSS stages:$\{1, 2\}$, #Params: 10.3M, FLOPs: 2.0G.
- Small – block:$\{2, 2, 5, 2\}$, QuadVSS stages:$\{1, 2\}$, #Params: 31.2M, FLOPs: 5.5G.
- Base – block:$\{2, 2, 15, 2\}$, QuadVSS stages:$\{1, 2\}$, #Params: 50.6M, FLOPs: 9.3G.

Experiment

Image classification results on ImageNet-1k

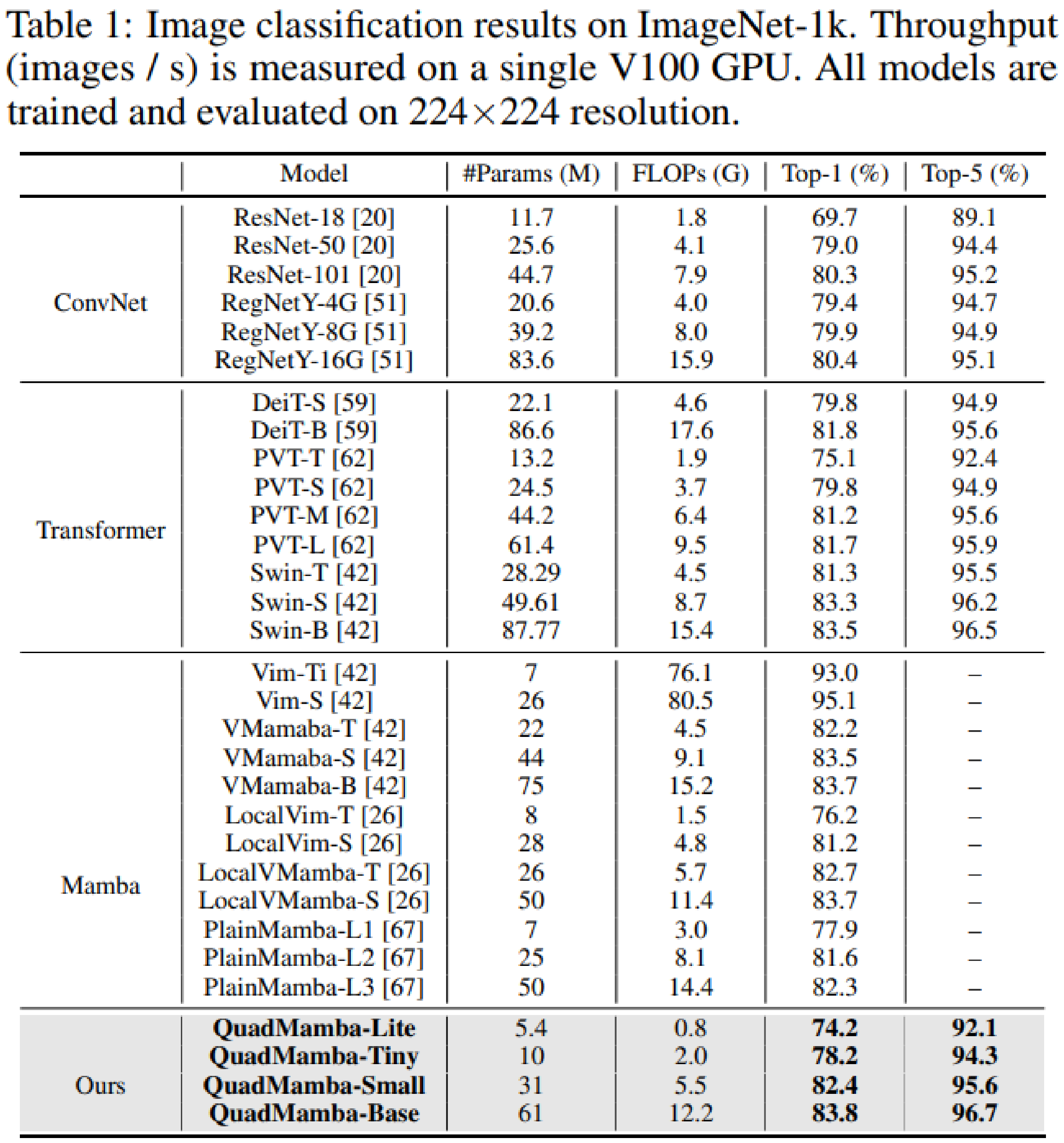


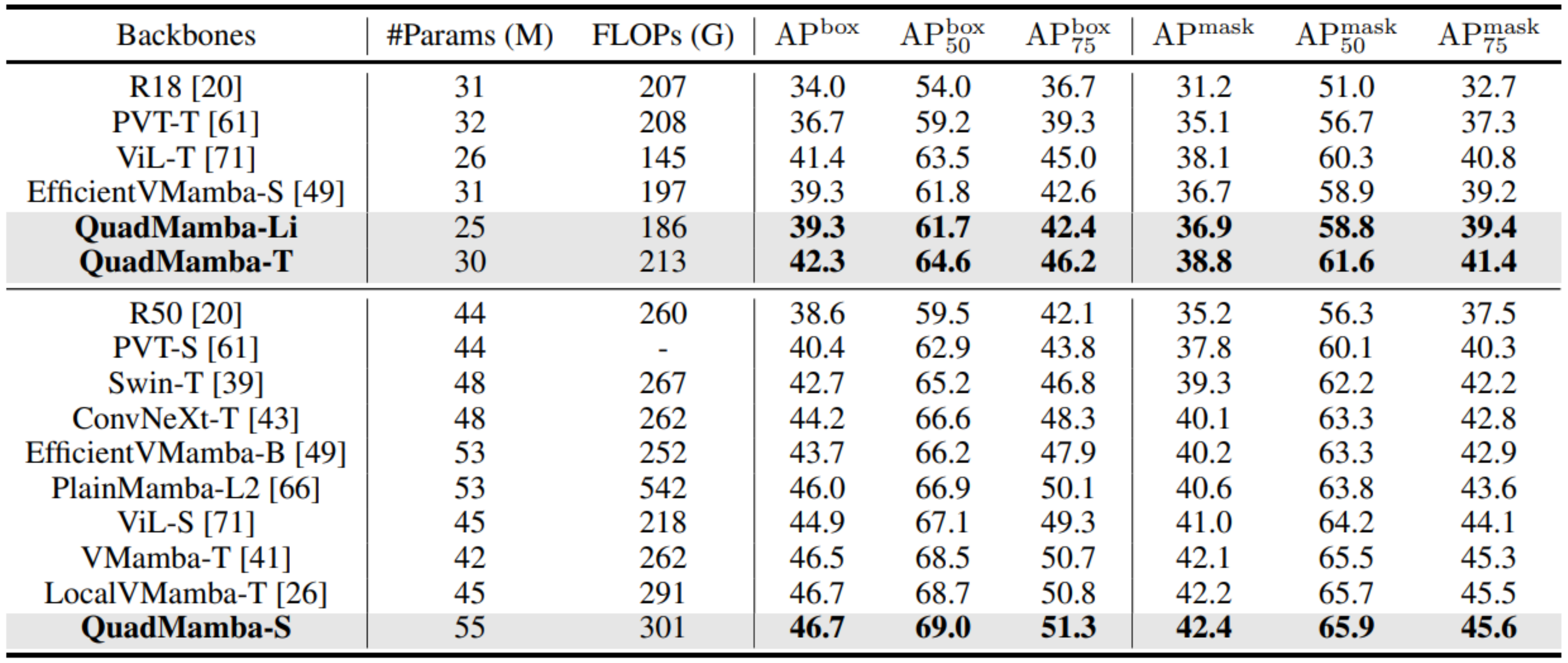
Table 1: Image classification results on ImageNet-1k. Throughput (images / s) is measured on a single V100 GPU. All models are trained and evaluated on 224×224 resolution.

	Model	#Params (M)	FLOPs (G)	Top-1 (%)	Top-5 (%)
ConvNet	ResNet-18 [20]	11.7	1.8	69.7	89.1
	ResNet-50 [20]	25.6	4.1	79.0	94.4
	ResNet-101 [20]	44.7	7.9	80.3	95.2
	RegNetY-4G [51]	20.6	4.0	79.4	94.7
	RegNetY-8G [51]	39.2	8.0	79.9	94.9
	RegNetY-16G [51]	83.6	15.9	80.4	95.1
Transformer	DeiT-S [59]	22.1	4.6	79.8	94.9
	DeiT-B [59]	86.6	17.6	81.8	95.6
	PVT-T [62]	13.2	1.9	75.1	92.4
	PVT-S [62]	24.5	3.7	79.8	94.9
	PVT-M [62]	44.2	6.4	81.2	95.6
	PVT-L [62]	61.4	9.5	81.7	95.9
	Swin-T [42]	28.29	4.5	81.3	95.5
	Swin-S [42]	49.61	8.7	83.3	96.2
	Swin-B [42]	87.77	15.4	83.5	96.5
Mamba	Vim-Ti [42]	7	76.1	93.0	–
	Vim-S [42]	26	80.5	95.1	–
	VMamaba-T [42]	22	4.5	82.2	–
	VMamaba-S [42]	44	9.1	83.5	–
	VMamaba-B [42]	75	15.2	83.7	–
	LocalVim-T [26]	8	1.5	76.2	–
	LocalVim-S [26]	28	4.8	81.2	–
	LocalVMamba-T [26]	26	5.7	82.7	–
	LocalVMamba-S [26]	50	11.4	83.7	–
	PlainMamba-L1 [67]	7	3.0	77.9	–
	PlainMamba-L2 [67]	25	8.1	81.6	–
	PlainMamba-L3 [67]	50	14.4	82.3	–
Ours	**QuadMamba-Lite**	5.4	0.8	**74.2**	**92.1**
	QuadMamba-Tiny	10	2.0	**78.2**	**94.3**
	QuadMamba-Small	31	5.5	**82.4**	**95.6**
	QuadMamba-Base	61	12.2	**83.8**	**96.7**

Experiment

Object detection and instance segmentation results on the COCO val2017.

Table 2: Object detection and instance segmentation results on the COCO val2017 split using the Mask RCNN [19] framework.

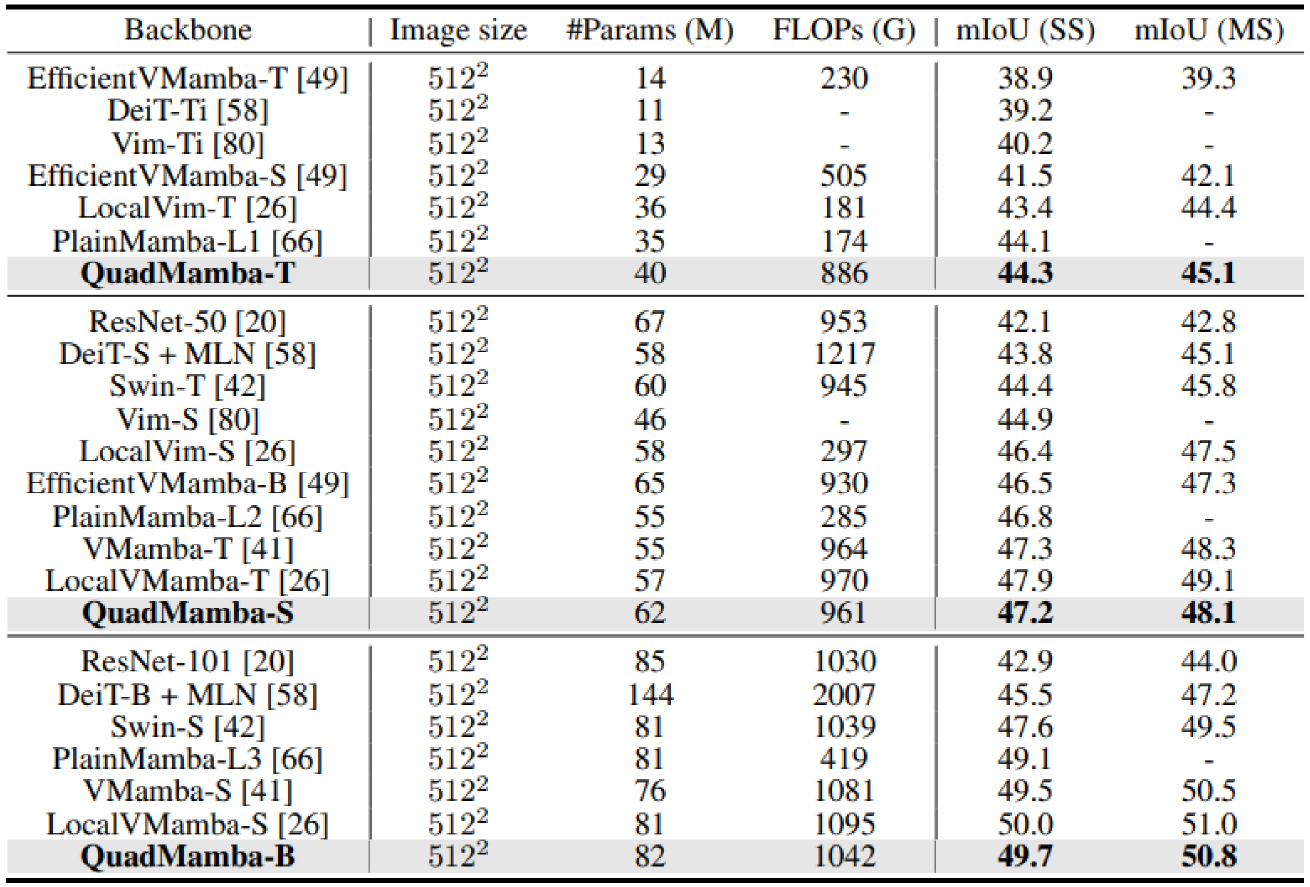


Backbones	#Params (M)	FLOPs (G)	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R18 [20]	31	207	34.0	54.0	36.7	31.2	51.0	32.7
PVT-T [61]	32	208	36.7	59.2	39.3	35.1	56.7	37.3
ViL-T [71]	26	145	41.4	63.5	45.0	38.1	60.3	40.8
EfficientVMamba-S [49]	31	197	39.3	61.8	42.6	36.7	58.9	39.2
QuadMamba-Li	25	186	**39.3**	**61.7**	**42.4**	**36.9**	**58.8**	**39.4**
QuadMamba-T	30	213	**42.3**	**64.6**	**46.2**	**38.8**	**61.6**	**41.4**
R50 [20]	44	260	38.6	59.5	42.1	35.2	56.3	37.5
PVT-S [61]	44	-	40.4	62.9	43.8	37.8	60.1	40.3
Swin-T [39]	48	267	42.7	65.2	46.8	39.3	62.2	42.2
ConvNeXt-T [43]	48	262	44.2	66.6	48.3	40.1	63.3	42.8
EfficientVMamba-B [49]	53	252	43.7	66.2	47.9	40.2	63.3	42.9
PlainMamba-L2 [66]	53	542	46.0	66.9	50.1	40.6	63.8	43.6
ViL-S [71]	45	218	44.9	67.1	49.3	41.0	64.2	44.1
VMamba-T [41]	42	262	46.5	68.5	50.7	42.1	65.5	45.3
LocalVMamba-T [26]	45	291	46.7	68.7	50.8	42.2	65.7	45.5
QuadMamba-S	55	301	**46.7**	**69.0**	**51.3**	**42.4**	**65.9**	**45.6**

Experiment

Semantic segmentation results on ADE20K.

Table 3: Semantic segmentation results on ADE20K using UperNet [62]. mIoUs are measured with single-scale (SS) and multi-scale (MS) testings on the *val* set. FLOPs are measured with an input size of 512×2048.



Backbone	Image size	#Params (M)	FLOPs (G)	mIoU (SS)	mIoU (MS)
EfficientVMamba-T [49]	512^2	14	230	38.9	39.3
DeiT-Ti [58]	512^2	11	-	39.2	-
Vim-Ti [80]	512^2	13	-	40.2	-
EfficientVMamba-S [49]	512^2	29	505	41.5	42.1
LocalVim-T [26]	512^2	36	181	43.4	44.4
PlainMamba-L1 [66]	512^2	35	174	44.1	-
QuadMamba-T	512^2	40	886	**44.3**	**45.1**
ResNet-50 [20]	512^2	67	953	42.1	42.8
DeiT-S + MLN [58]	512^2	58	1217	43.8	45.1
Swin-T [42]	512^2	60	945	44.4	45.8
Vim-S [80]	512^2	46	-	44.9	-
LocalVim-S [26]	512^2	58	297	46.4	47.5
EfficientVMamba-B [49]	512^2	65	930	46.5	47.3
PlainMamba-L2 [66]	512^2	55	285	46.8	-
VMamba-T [41]	512^2	55	964	47.3	48.3
LocalVMamba-T [26]	512^2	57	970	47.9	49.1
QuadMamba-S	512^2	62	961	**47.2**	**48.1**
ResNet-101 [20]	512^2	85	1030	42.9	44.0
DeiT-B + MLN [58]	512^2	144	2007	45.5	47.2
Swin-S [42]	512^2	81	1039	47.6	49.5
PlainMamba-L3 [66]	512^2	81	419	49.1	-
VMamba-S [41]	512^2	76	1081	49.5	50.5
LocalVMamba-S [26]	512^2	81	1095	50.0	51.0
QuadMamba-B	512^2	82	1042	**49.7**	**50.8**

Experiment-ablation

Model	W-Size	Top-1 (%)	AP^{box}	AP^{mask}
Lite	None	72.2	33.1	30.5
	28×28	72.9	33.8	31.7
	14×14	73.5	35.8	32.1
	2×2	72.4	33.4	30.6

Table 4: Impact of different local window sizes (W-Size) and using the naive flatten strategy.

Coarse	Fine	Top-1 (%)
(H, W)	$(\frac{H}{2}, \frac{W}{2})$	73.5
$(\frac{H}{2}, \frac{W}{2})$	$(\frac{H}{4}, \frac{W}{4})$	73.8
$(\frac{H}{4}, \frac{W}{4})$	$(\frac{H}{8}, \frac{W}{8})$	73.6

Table 5: Impact of partition resolution in QuadMamba.

Depths	#Params.	FLOPs	Top-1 (%)
2-2-8-2	31	9.2	82.0
2-8-2-2	38	8.1	81.5
2-2-2-8	74	7.8	81.8
2-4-6-2	36	8.5	82.1

Table 6: Impact of different depths with a channel dimension 96.

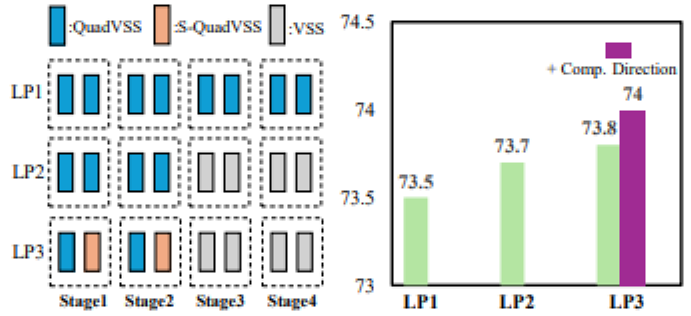


Figure 5: Impact of different QuadVSS layer patterns and shift directions.

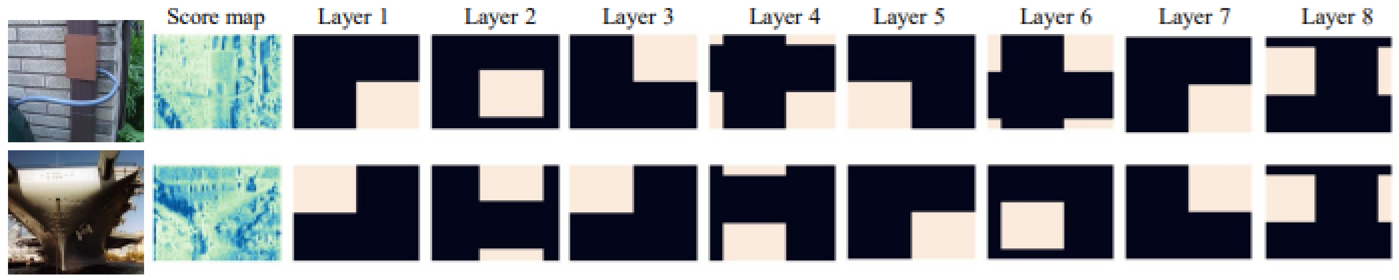


Figure 6: Visuliazation of partition maps which focus on different regions from shallow to deep blocks.

Conclusion

In this paper, we propose QuadMamba, a vision Mamba architecture that serves as a versatile and efficient backbone for visual tasks, such as image classification and dense predictions. QuadMamba effectively captures local dependencies of different granularities by learnable quadtree-based scanning, which adaptively preserves the inherent locality within image data with negligible computational overheads.

https://github.com/VISION-SJTU/QuadMamba