PVMamba: Parallelizing Vision Mamba via Dynamic State Aggregation

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

1. Overview of Appendix

In the appendix, we first provide additional architecture details in Sec. 2. Then, we present more theoretical analysis and details about the DSA scheme in Sec. 3. We provide the detailed experimental settings in Sec. 4. We also conduct more experiments in downstream tasks, including generative tasks in Sec. 5. In the final section, we provide some visualization results on generative tasks in Sec. 6.

2. Architecture Details

The detailed architecture specifications are presented in Tab. 1, assuming an input image size of 224×224 for all architecture variants. We also present a schematic view of the proposed DSA layers with a detailed illustration of the repeated pattern in Sa-DSA layers. "ConL (a, b, k3)" indicates a convolutional layer with a kernel size of 3×3 followed by the BN and ReLU layers. a and b indicate the input and output channel dimensions, respectively. In Fig. 1, we show the overall model structure of PVMamba with the detailed stacked layers in each model stage. We also present the layer pattern of the stacked Sa-DSA layers on the main stage. The repeated pattern is two reusing layers and one refining layer. The main model stage is repeated by stacking the Sa-DSA using an interleaved design.

3. Detail of DSA Scheme

3.1. Theoretical Analysis of SSM

State Space Models (S4). State Space Models (SSMs) are a general family of sequence models used in deep learning that are influenced by systems capable of mapping one-dimensional sequences in a continuous manner. These models transform input D-dimensional sequence $x(t) \in \mathbb{R}^{L\times D}$ into output sequence $y(t) \in \mathbb{R}^{L\times D}$ by utilizing a learnable latent state $h(t) \in \mathbb{R}^{N\times D}$ that is not directly observable. The mapping process could be denoted as:

$$\begin{aligned} h'(t) &= \boldsymbol{A}h(t) + \boldsymbol{B}x(t), \\ y(t) &= \boldsymbol{C}h(t), \end{aligned} \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{D\times N}$ and $\boldsymbol{C} \in \mathbb{R}^{D\times N}$.

Discretization. Discretization aims to convert the continuous differential equations into discrete functions, aligning the model to the input signal's sampling frequency for more efficient computation [8]. Following the work [9], the continuous parameters ($\boldsymbol{A}$, $\boldsymbol{B}$) can be discretized by zero-order hold rule with a given sample timescale $\Delta \in \mathbb{R}^{D}$:

$$\begin{aligned} \bar{\boldsymbol{A}} &= e^{\Delta \boldsymbol{A}}, \\ \bar{\boldsymbol{B}} &= (e^{\Delta \boldsymbol{A}} - \boldsymbol{I})\boldsymbol{A}^{-1}\boldsymbol{B}, \\ \bar{\boldsymbol{C}} &= \boldsymbol{C}, \\ \bar{\boldsymbol{B}} &\approx (\Delta \boldsymbol{A})(\Delta \boldsymbol{A})^{-1}\boldsymbol{A}\boldsymbol{B} = \Delta \boldsymbol{B}, \\ h(t) &= \bar{\boldsymbol{A}}h(t-1) + \bar{\boldsymbol{B}}x(t), \\ y(t) &= \bar{\boldsymbol{C}}h(t), \end{aligned} \tag{2}$$

where $\bar{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$, $\bar{\boldsymbol{B}} \in \mathbb{R}^{D\times N}$ and $\bar{\boldsymbol{C}} \in \mathbb{R}^{D\times N}$.

To simplify calculations, the repeated application of Equation 2 can be efficiently performed simultaneously using a global convolution approach.

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{x} \circledast \overline{\boldsymbol{K}} \\ \text{with} \quad \overline{\boldsymbol{K}} &= (\boldsymbol{C}\overline{\boldsymbol{B}}, \boldsymbol{C}\overline{\boldsymbol{A}\boldsymbol{B}}, ..., \boldsymbol{C}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}), \end{aligned} \tag{3}$$

where $\circledast$ denotes convolution operation, and $\overline{\boldsymbol{K}} \in \mathbb{R}^{L}$ is the SSM kernel.

Selective State Space Models (S6). Mamba [7] improves the performance of SSM by introducing Selective State Space Models (S6), allowing the continuous parameters to vary with the input and enhancing selective information processing across sequences, which extends the discretization process by selection mechanism:

$$\begin{aligned} \bar{\boldsymbol{B}} &= s_{\boldsymbol{B}}(x), \\ \bar{\boldsymbol{C}} &= s_{\boldsymbol{C}}(x), \\ \Delta &= \tau_{\boldsymbol{A}}(\text{Parameter} + s_{\boldsymbol{A}}(x)), \end{aligned} \tag{4}$$

* Corresponding author.

	downsp. rate (output size)	PVMamba-v1 (for ablation)	PVMamba-Tiny	PVMamba-Small	PVMamba-Base
stage 1	4× (56×56)	ConvL(3, 48, k3) ×3, [Local DSA sz.3 × 3, dim 48] × 2	ConvL(3, 64, k3) [Local DSA sz.3 × 3, dim 64] × 2	ConvL(3, 64, k3) [Local DSA sz.3 × 3, dim 64] × 3	ConvL(3, 96, k3) [Local DSA sz.3 × 3, dim 96] × 3
stage 2	8× (28×28)	ConvL(48, 96, k3) ×3, [Local DSA sz.3 × 3, dim 96] × 2	ConvL(64, 128, k3) [Local DSA sz.3 × 3, dim 192] × 4	ConvL(64, 128, k3) [Local DSA sz.3 × 3, dim 192] × 4	ConvL(96, 192, k3) [Local DSA sz.3 × 3, dim 384] × 4
stage 3	16× (14×14)	ConvL(96, 192, k3) ×3, [Sa-DSA Points.5, dim 96] × 8	ConvL(128, 256, k3) [Sa-DSA Points.5, dim 192] × 8	ConvL(128, 256, k3) [Sa-DSA Points.5, dim 192] × 21	ConvL(192, 384, k3) [Sa-DSA Points.5, dim 384] × 21
stage 4	32× (7×7)	ConvL(192, 384, k3) ×3, [Vanilla attention dim 384, head 24] × 4	ConvL(256, 512, k3) [Vanilla attention dim 512, head 24] × 4	ConvL(256, 512, k3) [Vanilla attention dim 512, head 24] × 5	ConvL(384, 768, k3) [Vanilla attention dim 768, head 24] × 5

Table 1. Detailed architecture specifications.

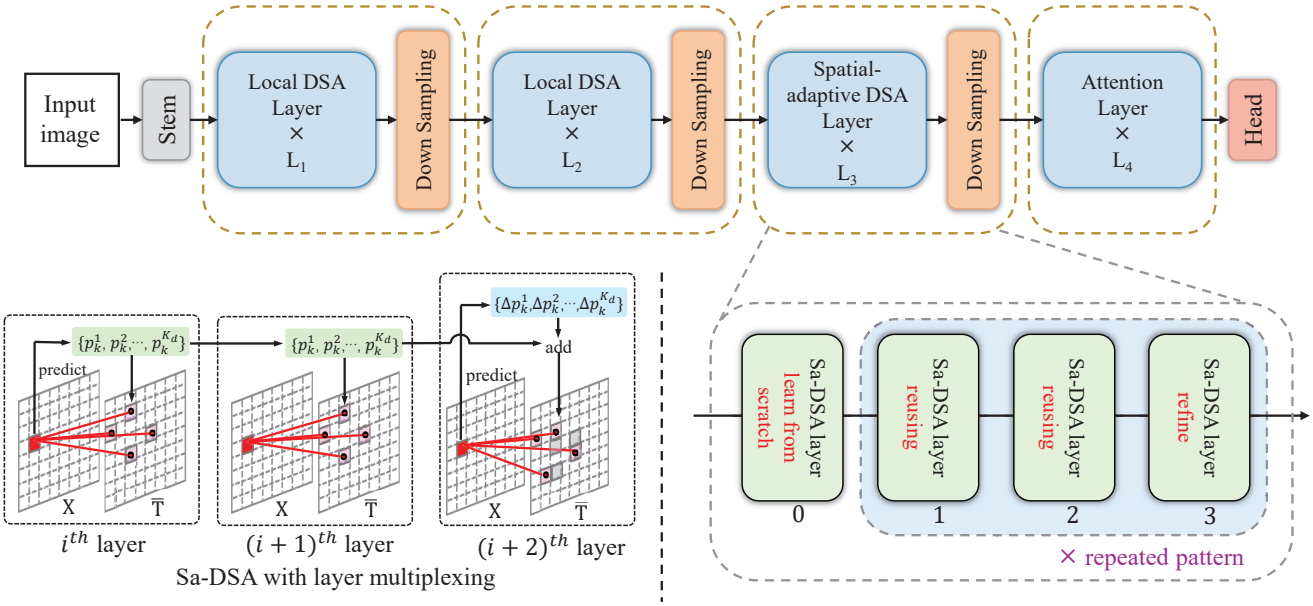

Figure 1. Detailed architecture of PVMamba and the layer multiplexing pattern.

where $s_B(x)$ and $s_C(x)$ are linear functions that project input x into an N-dimensional space, while $s_A(x)$ broadens a D-dimensional linear projection to the necessary dimensions.

3.2. Motivation to develop DSA scheme.

We present the pseudo-code of Mamba [7] scheme in Alg. 1. The for-loop procedure of solving SSM heavily relies on hardware optimization tricks to speed up and address the inherent sequential constraints when handling 2D visual data. Vision Mamba needs to convert 2D visual data into 1D sequential formats, limiting its spatial modeling capabilities. A sequential computing pipeline introduces latency, which is unfriendly to the current parallel deep learning paradigm.

To break the sequential constraint, we parallelize the sequential for-loop solving procedure using reasonable assumptions. We reformulate the original sequential computing in the Mamba scheme by matrix multiplication and a dynamic operator. The DSA scheme overcomes the sequential constraints that can directly process 2D visual data. The dynamic operator can maintain the linear computational complexity of the original Mamba. Our DSA scheme can dynamically select sparse points to model the spatial contexts using the SSM scheme. We assume that the hidden state can be parallelly aggregated by a selection of input pixel embedding x. Thus, we do not need to compute the hidden state recursively from the previous hidden state. Our assumption is based on the observation that the information density in

Algorithm 1 Pseudo code of Mamba scheme

```
#input: x; params: delta, A, B, C, D; output: out
def Mamba-SSM-Solver(x):
   # Generating Parameter Matrix for SSM
   delta = SoftPlus(Linear(x))
   B, C = Split(Linear(x))
   A, D = self.embedding_A, self.embedding_D
   deltaA = torch.exp(delta * A)
   deltaB_x = delta * B * x

   # For-loop solving procedure of SSM
   x = torch.zeros_like(A)
   ys = []
   for i in range(len(x)):
     x = deltaA[:, :, i, :] * x + deltaB_x[:, :, i
         , :]
     y = x * C[:, :, : , i]
     ys.append(y)
   y = torch.stack(ys)
   out = y + x * D
   return out
```

visual signals is far more sparse than the language [11], so the visual modeling is insensitive to the sequential order.

4. Detailed Experimental Settings

4.1. Image Classification

The training settings mostly follow [24] and [17]. For all model variants, we use a default input image resolution of 224 x 224 pixels. For different resolutions, such as 384 x 384 pixels, we fine-tune the models that were initially trained at the 224 x 224 resolution rather than training from scratch. This approach is based on the work presented in [17], and it helps to reduce GPU usage.

When training from scratch with a 224^2 input, we use the AdamW optimizer [18] for 300 epochs, employing a cosine decay learning rate scheduler with a linear warm-up over the first 20 epochs. The training setup includes a batch size of 1024, an initial learning rate of 0.001, a weight decay of 0.05, and gradient clipping with a maximum norm of 1. In our training process, we incorporate most of the augmentation and regularization strategies outlined in [24]. These include RandAugment [6], Mixup [28], CutMix [27], random erasing [29], and stochastic depth [15]. However, we do not use repeated augmentation [13] or Exponential Moving Average (EMA) [20], as these do not improve performance. It is important to note that this approach contrasts with the findings in [24].

4.2. Object Detection and Instance Segmentation

We consider a typical object detection framework: Cascade Mask R-CNN [1, 10] in mmdetection [3]. For detection task setting, we utilize the same settings: multi-scale training [2, 23] (resizing the input such that the shorter side is between 480 and 800 while the longer side is at most 1333), AdamW [18] optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs with the learning rate decayed by $10\times$ at epochs 27 and 33).

Method	Parameter	FID↓
DiT-S/8 [19]	33.1 M	151.2
PVMamba-S/8	**32.0** M	**136.4**

Table 2. Benchmarking class-conditional image generation on ImageNet 256×256. PVMamba-S/8 achieves state-of-the-art FID.

4.3. Semantic Segmentation

The ADE20K dataset [30] is a popular semantic segmentation dataset that includes a diverse array of 150 semantic categories. It contains a total of 25,000 images, with 20,000 designated for training, 2,000 for validation, and 3,000 for testing. For our implementation, we utilize UperNet [25] within the mmsegmentation framework [4], which was selected for its high efficiency.

For a fair comparison, we adopt a similar training setting following [16, 17]. During training, we utilize the AdamW optimizer [18], starting with an initial learning rate of 6×10^{-5} and a weight decay of 0.01. We implement a scheduler that employs linear learning rate decay along with a linear warmup for the first 1,500 iterations. The models are trained on 8 GPUs, with two images processed per GPU over a total of 160,000 iterations. For data augmentation, we follow the default settings in mmsegmentation, which include random horizontal flipping, random re-scaling within the range of [0.5, 2.0], and random photometric distortion.

During inference, a multi-scale test is conducted using resolutions that are 0.5, 0.75, 1.0, 1.25, 1.5, and 1.75 times the resolution used during training. When reporting test scores, both the training and validation images are included, following common practice [26].

5. Additional Experiments

5.1. On Generative Tasks

In recent years, generative tasks have become increasingly popular in the field of computer vision. These tasks involve the creation of new content using algorithms that can generate images, videos, or other types of visual media based on learned patterns from existing data. Generative tasks have diverse applications in vision, ranging from enhancing image resolution and style transfer to creating entirely new images that reflect different artistic styles or real-world scenes. These techniques are also being used in areas such as virtual reality, gaming, and even scientific fields like medicine,

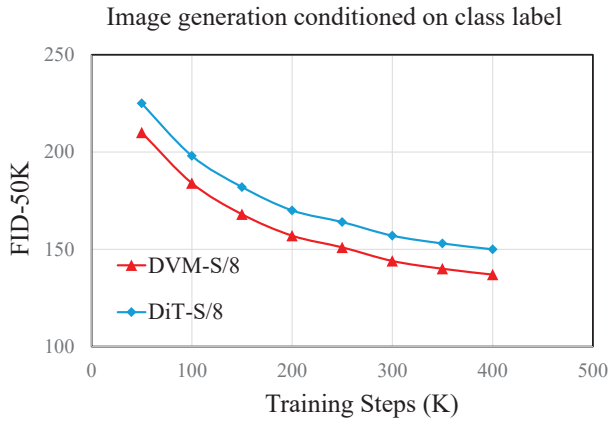

Figure 2. FID-50K results over training iterations. DVM refers to the model in which PVMamba is adopted as the latent diffusion transformer. The training curve of DiT [19] is also presented.

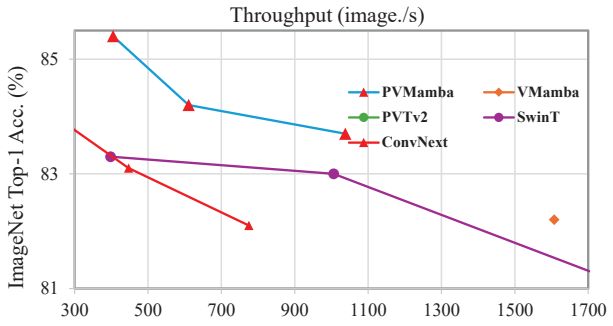

Figure 3. Throughput comparisons with other state-of-the-art methods.

where they can assist in synthesizing medical images for training and research purposes.

The combination of diffusion models [5] and several basic deep network architectures has proven successful in high-fidelity image and video generation tasks. We apply our DSA layer into the DiT-like [19] diffusion models to test the performance of generative tasks. The training settings mostly follow the DiT [19]. We show the qualitative and quantitative results using ImageNet benchmark [22]. As shown in Tab. 2 and Fig. 2, we measure the performance for the generative task by Frechet Inception Distance (FID) [12], the standard metric for evaluating generative models of images. Our PVMamba-S/8 model, which uses fewer model parameters (32.0M vs.33.1M) and computational requirements measured in FLOPs, outperforms the strong method DiT-S/8 by 14.8 in FID in the class-conditional image generation task. This success highlights not only the effectiveness of the model in generating high-quality images but also its potential for application in various generative downstream tasks [14, 19, 21].

The competitive results achieved by PVMamba-S/8 suggest that it possesses a robust generalization ability, allowing it to adapt effectively to different scenarios and data variations. This adaptability is essential for tasks that require generating images or other media types based on diverse input conditions. Overall, the PVMamba-S/8 model represents a significant advancement in generative modeling, underscoring its promise for future applications in the field.

6. Visualization Results

In Fig. 4 and Fig. 5, we showcase the Diffusion models that utilize the proposed PVMamba backbones. These models exhibit a promising level of image quality, highlighting their effectiveness in generating high-fidelity images. We have included selected samples from our class-conditional PVMamba-S/8 models, which were trained on the ImageNet dataset [22]. The training focused on producing images at a resolution of 256×256 pixels. By presenting these samples, we illustrate the capabilities of our models in rendering detailed and coherent images, demonstrating the potential of PVMamba backbones in the context of diffusion processes.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 3

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of European Conference on Computer Vision*, 2020. 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint*, 2019. 3

[4] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 3

[5] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *TPAMI*, 45(9):10850–10869, 2023. 4

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 3

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *ICML*, 2024. 1, 2

[8] 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. 1

[9] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *NeurIPS*, 2022. 1

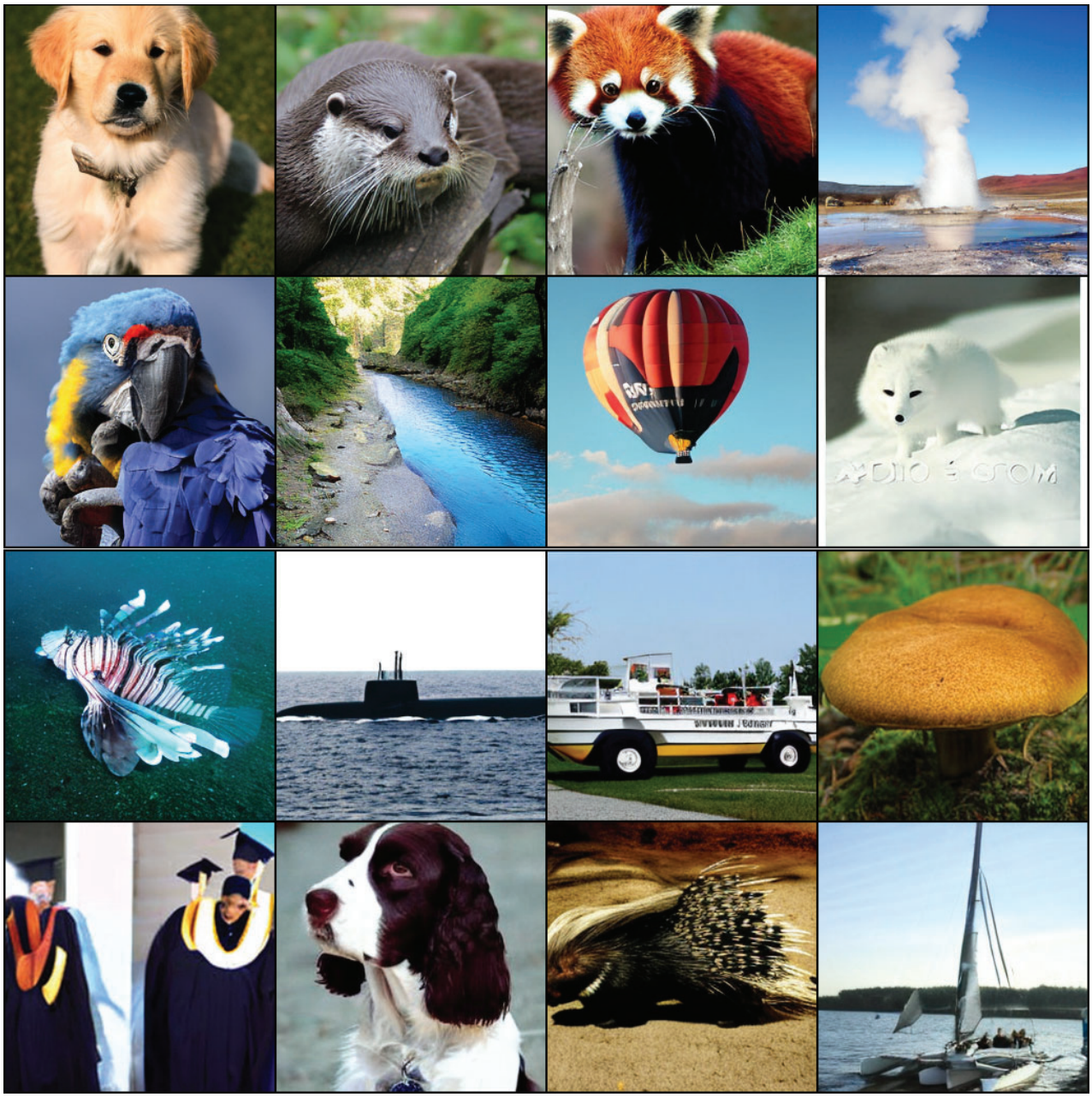

Figure 4. Visualization results of generative tasks with the proposed PVMamba backbone.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of International Conference on Computer Vision*, 2017. 3

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 4

[13] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[14] Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes Fischer, and Bjorn Ommer. Zigma: Zigzag mamba diffusion model. In *ECCV*,

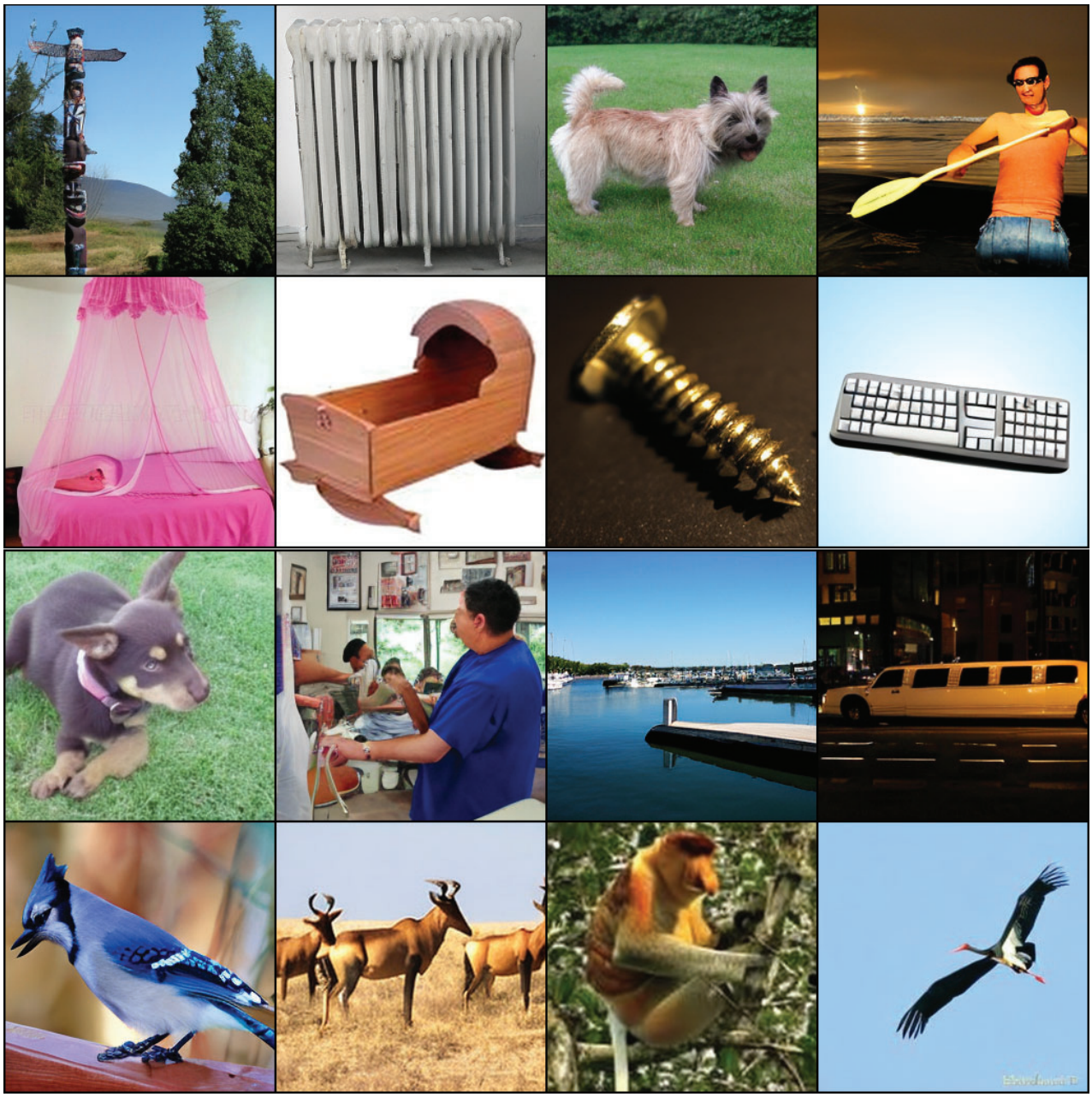

Figure 5. Visualization results of generative tasks with the proposed PVMamba backbone.

2024. 4

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proceedings of European Conference on Computer Vision*, 2016. 3

[16] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. In *NeurIPS*, 2024. 3

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 3

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 3

[19] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *CVPR*, 2023. 3, 4

[20] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *arXiv preprint*, 2019.

3

[21] Jiacheng Ruan and Suncheng Xiang. Vm-unet: Vision mamba unet for medical image segmentation. *arXiv:2402.02491*, 2024. 4

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large scale visual recognition challenge. *IJCV*, 115:211–252, 2015. 4

[23] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. *arXiv preprint*, 2020. 3

[24] Hugo Touvron, Matthieu Cord, Matthijs Douze, et al. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 3

[25] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 3

[26] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *ECCV*, 2020. 3

[27] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of International Conference on Computer Vision*, 2019. 3

[28] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, et al. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 3

[29] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *CoRR abs/1708.04896*, 2017. 3

[30] Bolei Zhou, Hang Zhao, Xavier Puig, et al. Scene parsing through ade20k dataset. In *CVPR*, 2017. 3