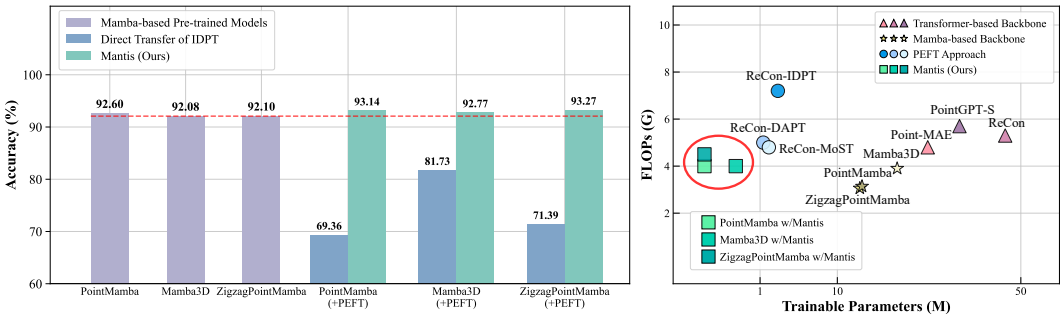
MANTIS: MAMBA-NATIVE TUNING IS EFFICIENT FOR 3D POINT CLOUD FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained 3D point cloud foundation models (PFMs) have demonstrated strong transferability across diverse downstream tasks. However, full fine-tuning these models is computationally expensive and storage-intensive. Parameter-efficient fine-tuning (PEFT) offers a promising alternative, but existing PEFT approaches are primarily designed for Transformer-based backbones and rely on token-level prompting or feature transformation. Mamba-based backbones introduce a granularity mismatch between token-level adaptation and state-level sequence dynamics. Consequently, straightforward transfer of existing PEFT approaches to frozen Mamba backbones leads to substantial accuracy degradation and unstable optimization. To address this issue, we propose Mantis, the first Mamba-native PEFT framework for 3D PFMs. Specifically, a State-Aware Adapter (SAA) is introduced to inject lightweight task-conditioned control signals into selective state-space updates, enabling state-level adaptation while keeping the pre-trained backbone frozen. Moreover, different valid point cloud serializations are regularized by Dual-Serialization Consistency Distillation (DSCD), thereby reducing serialization-induced instability. Extensive experiments across multiple benchmarks demonstrate that Mantis achieves competitive performance while training less than 5% of the backbone parameters. Our code is available in the supplementary material.



(a) Performance comparison (b) Efficiency comparison

Figure 1: Comprehensive comparisons between Mantis and several representative counterparts on OBJ_ONLY variant of ScanObjectNN dataset. (a) Mantis achieves superior performance over existing counterparts. (b) Our Mantis achieves an ideal trade-off between trainable parameters and FLOPs.

1 INTRODUCTION

Fine-tuning pre-trained point cloud foundation models (PFMs) has been demonstrated to adapt effectively to diverse downstream tasks(Yu et al., 2022; Pang et al., 2022; Zhang et al., 2022; Chen et al., 2023; Zha et al., 2024). However, the computational and storage costs associated with full fine-tuning are prohibitively high, particularly as model sizes continue to grow(Houlsby et al., 2019; Hu et al., 2022; Tang et al., 2024; Zhou et al., 2024; Chen et al., 2023; Qi et al., 2023). To alleviate this burden, parameter-efficient fine-tuning (PEFT) methods(Wang et al., 2025b; Liang et al., 2025; Zha et al., 2025a; Han et al., 2025; Zhou et al., 2024), such as Instance-aware Dynamic Prompt Tuning (IDPT)(Zha et al., 2023), have emerged as promising alternatives, garnering significant attention. Instead of updating all model parameters directly, IDPT applies DGCNN(Wang et al., 2019) to extract dynamic prompts for different instances. This approach significantly reduces the number of parameters that need to be fine-tuned, thereby lowering the computational and memory overhead.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Follow-up research in 3D point cloud analysis can be roughly categorized into Prompt Tuning(Zha et al., 2023; Ai et al., 2025), Adapters(Zhou et al., 2024; Zha et al., 2025b;a; Tang et al., 2024; Liang et al., 2025) and Reparameterization(Han et al., 2025; Wang et al., 2025b). Despite these advancements, most of the prior art still relies on Transformer-native architectures(Pang et al., 2022; Sun et al., 2026b). However, Transformer-based models incur quadratic token-length complexity and significant resource overhead, especially for large-scale point clouds and fine-grained tokenization. In parallel, recent advances in State Space Models (SSMs), particularly Mamba, offer a linear-time alternative to Transformer-based modeling(Gu et al., 2022; Gu & Dao, 2024). This advantage has already motivated a growing body of work on Mamba-based point cloud backbones, including PointMamba(Liang et al., 2024) and related variants(Han et al., 2024; Zhang et al., 2025).

Unfortunately, to the best of our knowledge, there is still no dedicated Mamba-native PEFT framework designed for 3D point cloud backbones. More critically, directly transferring existing PEFT approaches to frozen Mamba-based backbones leads to substantial accuracy degradation and severe optimization instability, as illustrated in Figure 1 (a). The catastrophic performance drop can be attributed to two fundamental mismatches. First, Transformer-native PEFT approaches operate at the token level, whereas Mamba models rely on state-level sequence dynamics, resulting in a granularity mismatch with frozen Mamba backbones. Second, point cloud Mamba models are sensitive to serialization-dependent state propagation, which is an inherent gap in the order-agnostic PEFT modules. This implies that indiscriminately applying existing PEFT approaches may undermine the potential of the Mamba architecture. This motivates the need for a Mamba-native PEFT framework.

In this paper, we propose Mantis, the first Mamba-native adaptation framework with two complementary components. State-Aware Adapter (SAA) integrates task-conditioned control signals into the Mamba state-space updates, bypassing token-level prompts and addressing the inherent granularity mismatch, while Dual-Serialization Consistency Distillation (DSCD) enforces consistent representations across two valid serializations of the same point cloud, regularizing order-sensitive state propagation and improving the preservation of pretrained geometric priors.

With the proposed components, Mantis achieves competitive performance across diverse downstream tasks, while training less than 5% of the backbone parameters. Specifically, Mantis with 0.8M parameters achieves 93.29%, 92.77%, and 93.48% in the three ScanObjectNN(Uy et al., 2019) variants on Mamba3D(Han et al., 2024), surpassing full fine-tuning with 16.9M parameters by **+ 0.17%, + 0.69% and + 1.43%**, respectively. Notably, by optimizing only lightweight task-specific modules, Mantis substantially reduces the trainable parameters and storage cost for downstream tasks, while preserving Mamba's linear-complexity modeling paradigm, as illustrated in Figure 1 (b). The main contributions of this work are summarized as follows:

- We reveal a fundamental gap between Transformer-native PEFT methods and Mamba-based backbones. To fill this gap, we propose Mantis, the first Mamba-native parameter-efficient fine-tuning framework for 3D point cloud analysis.
- Building upon our insights into the granularity mismatch, we propose **State-Aware Adapter (SAA)**, which injects task-conditioned control signals into Mamba state updates.
- Derived from the serialization-sensitive nature of Mamba modeling, we introduce **Dual-Serialization Consistency Distillation (DSCD)** for cross-serialization consistency and stable downstream optimization.
- Extensive experiments demonstrate the effectiveness and efficiency of our approach, shedding light on Mamba-native PEFT for future research.

2 Related work

2.1 Point cloud foundation models

PFMs are pre-trained on a pretext task, followed by fine-tuning for downstream applications. Architecturally, existing 3D PFMs can be divided into Transformer-based(Chen et al., 2023; Pang et al., 2022; Zha et al., 2024; Zhao et al., 2021; Wu et al., 2022; 2024) and Mamba-based(Liang et al., 2024; Han et al., 2024; Diao et al., 2025; Zhang et al., 2025) categories. Initially proposed for natural language processing(Vaswani et al., 2017; Devlin et al., 2019), Transformer-based approaches leverage self-attention to capture long-range dependencies and global context in 3D point cloud analysis.

For instance, Point-BERT(Yu et al., 2022), Point-MAE(Pang et al., 2022) and Point-M2AE(Zhang et al., 2022) simplify pre-training via masked autoencoding. Multimodal integration further enriches learned representations. ReCon(Qi et al., 2023) and HyperPoint(Sun et al., 2026a) embed contrastive objectives into generative pipelines to avoid overfitting in Transformer-based approaches.

Meanwhile, Mamba-based approaches have emerged as a new paradigm for PFMs, offering efficient alternatives to Transformers by relying on SSMs. PointMamba(Liang et al., 2024) pioneers point cloud modeling, providing strong global representations for downstream tasks, while Mamba3D(Han et al., 2024) focuses on enhancing local features. HydraMamba(Qu et al., 2025), StruMamba3D(Wang et al., 2025a) and OctMamba(Jiang et al., 2026) employ various space-filling curves to form causal sequences for SSM. In contrast to the above methods, PCM(Zhang et al., 2025) and CloudMamba(Qu et al., 2026) directly build structural dependencies without explicit curve parameterization. While these pre-trained 3D vision models with full fine-tuning have demonstrated exceptional capabilities for various downstream 3D tasks, the high computational costs and potential dilution of pre-trained knowledge motivate our exploration of efficient fine-tuning strategies.

2.2 Parameter-efficient fine-tuning

As highlighted above, PFMs are iterating at an unprecedented pace in terms of structure(Sun et al., 2026d), method(Wang et al., 2026; Li et al., 2025; Wu et al., 2025), and applications(Zhang et al., 2026). This momentum has rendered the PEFT community equally dynamic. PEFT in 3D point cloud approaches mainly fall into prompt-based(Zha et al., 2023; Ai et al., 2025), adapter-based(Zhou et al., 2024; Zha et al., 2025b;a; Tang et al., 2024; Liang et al., 2025) and reparameterization-based(Han et al., 2025; Wang et al., 2025b). Prompt-based approaches add learnable tokens to input data or attention layers to guide task-specific tuning. For example, IDPT(Zha et al., 2023) introduces instance-aware dynamic prompts to improve robustness during downstream adaptation. Meanwhile, adapter-based pioneers like DAPT(Zhou et al., 2024) combines dynamic adapter scaling with prompt generation to better match downstream token significance. In order to overcome the limitation in inference overhead, reparameterization-based approaches reformulate the network weights, enabling parameter-efficient adaptation. PointLoRA(Wang et al., 2025b) brings low-rank adaptation with local token priors to fine-tuning and MoST(Han et al., 2025) brings Monarch-based sparse reparameterization with local geometric priors. However, these tuning approaches are specifically designed for Transformer-based backbones, which limits their adaptability to Mamba-based models. In contrast, our Mantis, the first Mamba-native parameter-efficient fine-tuning framework for 3D point cloud analysis, fills the underexplored gap in PEFT for Mamba-based backbones and enables efficient adaptation to downstream tasks while preserving Mamba's linear-complexity modeling.

3 Methods

3.1 Overall pipeline

The pipeline of our approach is shown in Figure 2. Given an input point cloud $P \in \mathbb{R}^{M\times 3}$, we first utilize FPS to sample n representative key points, denoted as $p \in \mathbb{R}^{n\times 3}$. The order of key points p is generally random, which does not affect previous Transformer-based approaches due to their inherent order-invariance. However, for the selective SSM, its unidirectional modeling struggles with unstructured point clouds. To address this, building on the pioneer works(Liang et al., 2024), we leverage space-filling curves (*e.g.*, the Hilbert curve(Hilbert, 1935) and its transposed variant) for dual-order serialization, transforming the unstructured point clouds into regular sequences, denoted as p_h and $p_{h'}$. We then apply the KNN algorithm to select K nearest neighbors for each key point, forming n local token patches $\mathcal{N}_h^{(1)} \in \mathbb{R}^{n\times K\times 3}$ and $\mathcal{N}_{h'}^{(2)} \in \mathbb{R}^{n\times K\times 3}$ with patch size K. To map the local patches to feature space, we employ a lightweight PointNet(Qi et al., 2017a) to obtain serialized point tokens $E_h^{(1)} \in \mathbb{R}^{n\times d}$ and $E_{h'}^{(2)} \in \mathbb{R}^{n\times d}$.

Different from existing approaches that employ feature-wise affine transformations as order indicators(Liang et al., 2024), we model the serialization order as a conditioning signal that modulates the representation in a residual manner. Specifically, for each branch $k \in \{1, 2\}$, we introduce a learnable order embedding $o^{(k)} \in \mathbb{R}^{d_o}$. The order-aware representation $e^{(k)} \in \mathbb{R}^d$ is formulated as follows:

$$e^{(k)} = \mathrm{MaxPool}(E^{(k)} + \mathcal{G}\Big(E^{(k)}\Big) \odot \varphi\Big(o^{(k)}\Big)), \tag{1}$$

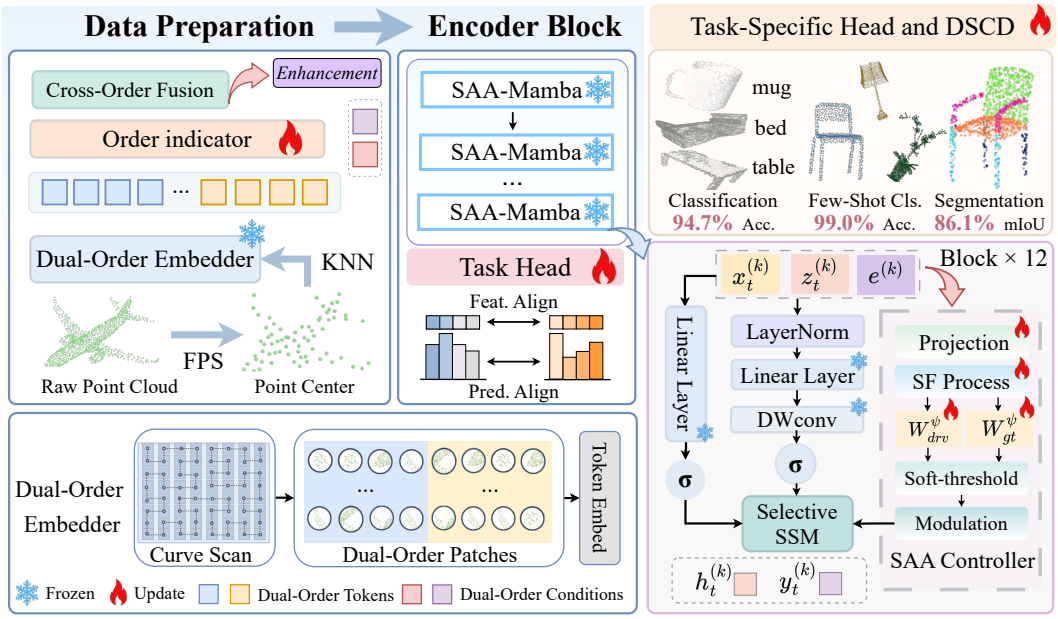


Figure 2: **Pipeline of Mantis.** Raw point clouds are processed by Farthest Point Sampling (FPS) and K-nearest neighbors (KNN) to generate local patches. The resulting patches are serialized along two complementary space-filling curves to produce dual-order patches. These patches are processed by stacked SAA-Mamba blocks, where the selective SSM is modulated by the SAA controller. DSCD regularizes cross-serialization consistency at both feature and prediction levels during training, while only a single serialization branch is retained at inference. During training, the backbone is frozen and only newly added parameters are fine-tuned. σ indicates SiLU activation.

where $\mathcal{G}(\cdot)$ denotes a lightweight transformation on the input features, and $\varphi(\cdot)$ maps the order embedding to a modulation vector in the feature space. $\odot$ denotes element-wise product. The two serialized sequences capture complementary structural cues under different traversal orders, while DSCD enforces cross-order consistency during training. The resulting representations $Z_0^{(k)} \in \mathbb{R}^{n\times d}$ are then fed into a non-hierarchical encoder composed of stacked SAA-Mamba blocks. Each SAA-Mamba block follows the general design of a Mamba layer, consisting of layer normalization, linear projection, depth-wise convolution, selective state-space modeling, and residual connection. Specifically, given the input $Z_{l-1}^{(k)}$ to the l-th block, we first obtain the intermediate features by:

$$\tilde{Z}_l^{(k)} = \mathrm{Linear}\left(\mathrm{LayerNorm}(Z_{l-1}^{(k)})\right), \quad X_l^{(k)} = \sigma\Big(\mathrm{DWConv}\left(\tilde{Z}_l^{(k)}\right)\Big), \tag{2}$$

where σ indicates SiLU activation. The resulting feature matrix $X_l^{(k)}$ is then fed into the selective SSM for sequence modeling. Different from standard Mamba blocks, we introduce a **State-Aware Adapter (SAA)** controller to modulate the state transition dynamics within the selective SSM. The output of the l-th block is computed as:

$$Z_l^{(k)} = \mathcal{F}_{\mathrm{SAA}}\left(X_l^{(k)}\right) + \sigma(\mathrm{Linear}(Z_{l-1}^{(k)})), \tag{3}$$

where $\mathcal{F}_{\mathrm{SAA}}(\cdot)$ denotes the selective SSM enhanced with the proposed State-Aware Adapter.

3.2 State-Aware Adapter

In the SAA-Mamba block described above, we follow the selective SSM formulation, where x_t, h_t, and y_t denote the input, hidden state, and output at sequence step t, respectively. The state dynamics are governed by the structured transition operator $\boldsymbol{A}$ and the input-dependent selective coefficients $\boldsymbol{B}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{\Delta}_t$, which control input injection, readout, and discretization, respectively. Detailed preliminaries are provided in Appendix A.1. Based on this observation, we introduce a lightweight, task-conditioned control signal to modulate the input-dependent selective coefficients, enabling

adaptation of the state evolution while preserving the structured transition operator. In contrast to LoRA-style static weight updates(Hu et al., 2022; Liu et al., 2024), our control signal is input-dependent and dynamically modulates the state evolution at each step. Moreover, SAA acts directly on state-space dynamics rather than inserting adaptation solely into token-level representations, providing fine-grained control over sequence evolution while keeping the backbone frozen. At each step t, the SAA controller takes the current SSM input $x_t^{(k)}$, the incoming block representation $z_t^{(k)}$, and the order-aware representation $e^{(k)}$ as inputs, where $z_t^{(k)}$ denotes the t-th token of $Z_{l-1}^{(k)}$. Since these quantities are available before the selective scan, the control signals for all sequence positions can be generated in parallel:

$$\phi_t^{(k)} = \Phi\Big(W_x^{\phi} x_t^{(k)} \,\|\, W_z^{\phi} z_t^{(k)} \,\|\, W_e^{\phi} e^{(k)}\Big), \tag{4}$$

where $\Phi(\cdot)$ denotes the control-feature fusion process, $\|$ denotes feature-wise concatenation, and $W_x^{\phi}, W_z^{\phi}, W_e^{\phi} \in \mathbb{R}^{d_\phi \times d}$ project the three pre-scan representations into a shared d_ϕ-dimensional control space. We formulate control signal generation as a proximal optimization problem, which admits a closed-form soft-thresholding solution. Specifically, denote $\mathbf{q}_t^{(k)} = W_{\mathrm{drv}}^{\psi}\, \phi_t^{(k)}$, and $\boldsymbol{\lambda}_t^{(k)} = -\log \mathrm{sigmoid}\Big(W_{\mathrm{gt}}^{\psi}\, \phi_t^{(k)}\Big)$. The optimal sparse control signal is obtained by minimizing:

$$\mathbf{u}_t^{(k)} = \arg\min_{\mathbf{v}} \frac{1}{2} \Big\| \mathbf{q}_t^{(k)} - \mathbf{v} \Big\|_2^2 + \sum_i \lambda_{t,i}^{(k)} \, |v_i| \,, \tag{5}$$

where v_i is the i-th entry of the optimization variable $\mathbf{v} \in \mathbb{R}^r$. Its closed-form solution is the component-wise thresholding operator:

$$u_{t,i}^{(k)} = \begin{cases} q_{t,i}^{(k)} - \lambda_{t,i}^{(k)}, & \text{if } q_{t,i}^{(k)} > \lambda_{t,i}^{(k)}, \\ 0, & \text{if } \Big| q_{t,i}^{(k)} \Big| \le \lambda_{t,i}^{(k)}, \\ q_{t,i}^{(k)} + \lambda_{t,i}^{(k)}, & \text{if } q_{t,i}^{(k)} < -\lambda_{t,i}^{(k)}, \end{cases} \tag{6}$$

where $q_{t,i}^{(k)}$ and $\lambda_{t,i}^{(k)}$ denote the i-th element of $\mathbf{q}_t^{(k)}$ and $\boldsymbol{\lambda}_t^{(k)}$, respectively. $W_{\mathrm{drv}}^{\psi}, W_{\mathrm{gt}}^{\psi} \in \mathbb{R}^{r \times d_\phi}$ are learnable projection matrices, and $r \ll d$ denotes the bottleneck dimension. To achieve task-conditioned adaptation, SAA directly modulates the native input-dependent selective coefficients $(\boldsymbol{B}_t^{(k)}, \boldsymbol{C}_t^{(k)}, \boldsymbol{\Delta}_t^{(k)})$ via the shared sparse control signal $\mathbf{u}_t^{(k)}$:

$$\widetilde{\boldsymbol{O}}_t^{(k)} = \boldsymbol{O}_t^{(k)} + \mathcal{R}_O\Big(\mathbf{U}_O \,\mathrm{diag}\Big(\mathbf{u}_t^{(k)}\Big)\, \mathbf{V}_O\Big), \qquad O \in \{B, C, \Delta\}. \tag{7}$$

where $\mathbf{U}_O$ and $\mathbf{V}_O$ are coefficient-specific low-rank modulation matrices, and $\mathcal{R}_O(\cdot)$ maps each factorized correction to the native coefficient layout. The structured transition operator $\boldsymbol{A}$ remains frozen. Since $\mathbf{u}_t^{(k)}$ depends only on pre-scan representations, all modulated coefficients are generated before recurrence and supplied to the backbone's native fused selective-scan operator. The structured discretization performed within the selective SSM can be written as:

$$\widehat{\boldsymbol{A}}_t^{(k)} = \exp\Big(\boldsymbol{A}\, \widetilde{\boldsymbol{\Delta}}_t^{(k)}\Big), \quad \widehat{\boldsymbol{B}}_t^{(k)} = \boldsymbol{A}^{-1}\Big(\exp\Big(\boldsymbol{A}\, \widetilde{\boldsymbol{\Delta}}_t^{(k)}\Big) - \boldsymbol{I}\Big)\, \widetilde{\boldsymbol{B}}_t^{(k)}. \tag{8}$$

Eq. (8) describes the corresponding structured discretization mathematically. In implementation, these quantities are evaluated through the native factorized selective-scan computation without explicitly materializing dense transition matrices or general matrix exponentials. As a consequence, SAA formulates the selective SSM as a conditionally controlled dynamical system while preserving its native recurrence. Importantly, all SAA coefficients are computed in parallel before a single native selective-scan call per SAA-Mamba block and serialization branch; no auxiliary recurrent pass or recurrent approximation is introduced. Appendix A.3 further analyzes the induced selective-kernel perturbation and hidden-state deviation.

3.3 Dual-Serialization Consistency Distillation

Although serialization enables Mamba to process unordered point clouds as sequential inputs, the resulting hidden-state propagation is inherently sensitive to the specific traversal order. Consequently, different valid serializations of the same point cloud may induce inconsistent feature representations and unstable predictions, which hinders reliable adaptation under frozen Mamba backbones. We provide a transfer-kernel analysis of this serialization sensitivity in Appendix A.2.

To alleviate this issue, we propose **Dual-Serialization Consistency Distillation (DSCD)**, which enforces cross-order consistency by aligning feature representations and predictive distributions across two serialization views, thereby reducing serialization-induced discrepancies, as illustrated in Figure 3. Unlike generic augmentation consistency, the two views preserve exactly the same underlying point geometry but induce distinct traversal-dependent state propagation patterns in Mamba. Built upon the dual-serialization pipeline described above, DSCD regularizes the two serialization branches at the final encoder layer. The final-layer output of branch k is denoted by $Z_L^{(k)}$, and the feature alignment loss is formulated as:

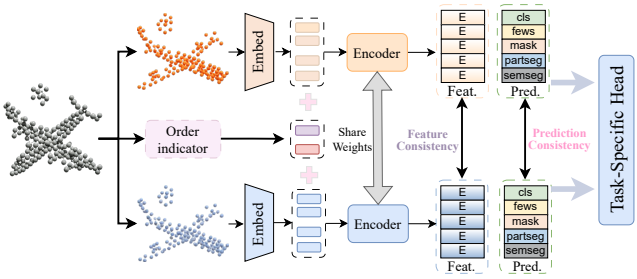


Figure 3: **Illustration of Dual-Serialization Consistency Distillation.** During training, DSCD enforces cross-serialization consistency at both feature and prediction levels.

$$z^{(k)} = g_\varphi\Big(\mathrm{AvgPool}(Z_L^{(k)})\Big), \qquad \mathcal{L}_{\mathrm{feat}} = \left\| \frac{z^{(1)}}{\|z^{(1)}\|_2} - \frac{z^{(2)}}{\|z^{(2)}\|_2} \right\|_2^2, \tag{9}$$

where g_φ is a branch-shared projection mapping. While $\mathcal{L}_{\mathrm{feat}}$ enforces representation-level consistency across different serialization orders, stable downstream optimization further requires alignment at the decision level. The prediction-level consistency term is defined as:

$$\pi^{(k)} = \mathrm{Softmax}\Big(\ell^{(k)}/\tau\Big), \qquad \mathcal{L}_{\mathrm{pred}} = \frac{1}{2}\,\mathrm{KL}\Big(\pi^{(1)} \,\|\, \pi^{(2)}\Big) + \frac{1}{2}\,\mathrm{KL}\Big(\pi^{(2)} \,\|\, \pi^{(1)}\Big), \tag{10}$$

where τ is the temperature hyperparameter, and $\pi^{(k)}$ denotes the softened predictive distribution of branch k. Accordingly, the training objective can be defined as follows:

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \alpha\,\mathcal{L}_{\mathrm{feat}} + \beta\,\mathcal{L}_{\mathrm{pred}}, \tag{11}$$

where α and β are coefficients for feature consistency and prediction consistency, respectively. By reducing cross-serialization discrepancies at both the representation and prediction levels, DSCD improves cross-serialization consistency and stabilizes downstream optimization.

4 Experiments

We evaluate Mantis on multiple downstream tasks, including object classification, part and semantic segmentation. We utilize three pre-trained models (PointMamba(Liang et al., 2024), Mamba3D(Han et al., 2024), and ZigzagPointMamba(Diao et al., 2025)) as our baselines. For a fair comparison, we follow the default fine-tuning settings of each baseline. The hyperparameters are set as $\alpha = 100$, $\beta = 0.05$, and $\tau = 1$. Additional analyses of α and β and training details can be found in Appendix B.

4.1 Object Classification

Real-World Object Classification. The ScanObjectNN(Uy et al., 2019) dataset is a highly challenging 3D dataset covering ∼15K real-world objects across 15 categories. As shown in Table 1, we conduct experiments on three variants of ScanObjectNN (*i.e.*, OBJ_BG, OBJ_ONLY, and PB_T50_RS), each with increasing complexity. Mantis surpasses full fine-tuning in most cases with only about 5% of the parameters. Specifically, Mantis surpasses the full fine-tuning of PointMamba, Mamba3D, and ZigzagPointMamba by 0.65%, 1.43%, 1.46% on PB_T50_RS split, respectively. Furthermore, the t-SNE feature visualizations(van der Maaten & Hinton, 2008) of existing methods and Mantis are presented in Figure 6 (a). This suggests that Mantis is especially effective in improving robustness under noisy and order-sensitive real-world point cloud distributions. Notably, the more pronounced performance improvements on the most challenging split indicate that dynamic state-aware adaptation and cross-serialization consistency become increasingly beneficial when serialization-induced instability is amplified. A detailed analysis of this phenomenon is provided in Appendix C.2.

Synthetic Object Classification. The ModelNet40(Wu et al., 2015) dataset contains a total of 12,311 3D CAD models across 40 categories. Due to the computational cost of the voting strategy(Liu et al., 2019), we report overall accuracy without voting. As shown in Table 1, compared to full fine-tuning, Mantis substantially reduces resource requirements while achieving comparable performance.

Table 1: Classification on three variants of ScanObjectNN(Uy et al., 2019) and ModelNet40(Wu et al., 2015), including the number of trainable parameters, inference FLOPs and overall accuracy (OA). #TP (M) represents the trainable parameters. Attn. denotes attention-based (Transformer) and SSM denotes state-space model (Mamba). † indicates that using simple rotational augmentation(Dong et al., 2023) for training. ○ and ○ denote methods that leverage cross-modal and single-modal information, respectively. ModelNet40 results report the overall accuracy from 1024 points without voting. Our results are averaged over three random seeds. * denotes results reproduced from the public source code. The best two results are bolded and underlined respectively. Same as follows.

Methods	Reference	Backbone	#TP (M)↓	FLOPs (G)	ScanObjectNN↑			ModelNet40↑
					OBJ_BG	OBJ_ONLY	PB_T50_RS	OA (%)
Supervised Learning Only								
○ PointNet(Qi et al., 2017a)	CVPR17	-	3.5	0.5	73.3	79.2	68.0	89.2
○ PointNet++(Qi et al., 2017b)	NeurIPS17	-	1.5	1.7	82.3	84.3	77.9	90.7
○ PointCNN(Li et al., 2018)	NeurIPS18	-	0.6	0.9	86.1	85.5	78.5	92.2
○ DGCNN(Wang et al., 2019)	TOG19	-	1.8	2.4	82.8	86.2	78.1	92.9
Self-Supervised Learning (Full Fine-tuning)								
○ Point-BERT(Yu et al., 2022)	CVPR22	Attn.	22.1	4.8	87.43	88.12	83.07	93.2
○ Point-M2AE(Zhang et al., 2022)	NeurIPS22	Attn.	15.3	3.6	91.22	88.81	86.43	94.0
○ PointGPT-S†(Chen et al., 2023)	NeurIPS23	Attn.	29.2	5.7	93.39	92.43	89.17	93.3
○ ACT†(Dong et al., 2023)	ICLR23	Attn.	22.1	4.8	93.29	91.91	88.21	93.6
○ Joint-MAE(Guo et al., 2023)	IJCAI23	Attn.	-	-	90.94	88.86	86.07	94.0
○ I2P-MAE†(Zhang et al., 2023)	CVPR23	Attn.	15.3	-	94.15	91.57	90.11	93.4
○ PCM(Zhang et al., 2025)	AAAI25	SSM	34.2	-	-	-	88.10	93.4
○ SI-Mamba(Bahri et al., 2025)	CVPR25	SSM	12.3	3.6	92.25	91.39	87.30	92.7
Self-Supervised Learning (Parameter-Efficient Fine-Tuning)								
○ Point-MAE(Pang et al., 2022)	ECCV22	Attn.	22.1 (100%)	4.8	90.02	88.29	85.18	93.8
+ IDPT(Zha et al., 2023)	ICCV23	Attn.	1.7 (7.7%)	7.2	91.22 (+ 1.20)	90.02 (+ 1.73)	84.94 (- 0.24)	93.3 (- 0.5)
+ PointLoRA(Wang et al., 2025b)	CVPR25	Attn.	0.8 (3.6%)	-	90.71 (+ 0.69)	89.33 (+ 1.04)	85.53 (+ 0.35)	93.3 (- 0.5)
+ PoinTriE*(Sun et al., 2026c)	ECCV26	Attn.	1.1 (5.0%)	4.2	91.30 (+ 1.28)	90.13 (+ 1.84)	85.71 (+ 0.53)	93.7 (- 0.1)
○ ReCon†(Qi et al., 2023)	ICML23	Attn.	43.6 (100%)	5.3	**95.18**	**93.63**	90.63	94.1
+ DAPT(Zhou et al., 2024)	CVPR24	Attn.	1.1 (2.5%)	5.0	94.32 (- 0.86)	92.43 (- 1.20)	89.38 (- 1.25)	93.5 (- 0.6)
+ GAPrompt(Ai et al., 2025)	ICML25	Attn.	0.6 (1.4%)	5.0	94.49 (- 0.69)	92.60 (- 1.03)	89.76 (- 0.87)	94.0 (- 0.1)
+ PoinTriE*(Sun et al., 2026c)	ECCV26	Attn.	0.8 (1.8%)	4.8	94.91 (- 0.27)	93.19 (- 0.44)	90.11 (- 0.52)	94.1 (+ 0.0)
○ PointMamba†(Liang et al., 2024)	NeurIPS24	SSM	12.3 (100%)	3.1	94.32	92.60	89.31	93.6
+ PMA*(Zha et al., 2025a)	CVPR25	SSM	1.1 (8.9%)	5.2	93.53 (- 0.79)	91.32 (- 1.28)	86.71 (- 2.60)	93.6 (+ 0.0)
+ PoinTriE*(Sun et al., 2026c)	ECCV26	SSM	0.8 (6.5%)	4.6	92.81 (- 1.51)	91.35 (- 1.25)	86.97 (- 2.34)	93.7 (+ 0.1)
+ Mantis†	**This Paper**	SSM	0.6 (4.9%)	4.0	94.65 (+ 0.33)	93.14 (+ 0.54)	89.96 (+ 0.65)	93.7 (+ 0.1)
○ Mamba3D(Han et al., 2024)	ACM MM24	SSM	16.9 (100%)	3.9	93.12	92.08	92.05	**94.7**
+ PMA*(Zha et al., 2025a)	CVPR25	SSM	1.3 (7.7%)	4.7	91.97 (- 1.15)	89.64 (- 2.44)	88.70 (- 3.35)	94.2 (- 0.5)
+ PoinTriE*(Sun et al., 2026c)	ECCV26	SSM	1.0 (5.9%)	4.6	92.11 (- 1.01)	90.03 (- 2.05)	88.82 (- 3.23)	94.4 (- 0.3)
+ Mantis	**This Paper**	SSM	0.8 (4.7%)	4.0	93.29 (+ 0.17)	92.77 (+ 0.69)	**93.48** (+ 1.43)	**94.7** (+ 0.0)
○ ZigzagPointMamba(Diao et al., 2025)	NeurIPS25	SSM	12.3 (100%)	3.1	94.15	92.10	88.65	93.2
+ PMA*(Zha et al., 2025a)	CVPR25	SSM	1.1 (8.9%)	5.7	93.10 (- 1.05)	89.53 (- 2.57)	85.18 (- 3.47)	92.8 (- 0.4)
+ PoinTriE*(Sun et al., 2026c)	ECCV26	SSM	0.8 (6.5%)	5.2	93.37 (- 0.78)	90.17 (- 1.93)	85.92 (- 2.73)	93.1 (- 0.1)
+ Mantis	**This Paper**	SSM	0.6 (4.9%)	4.5	94.02 (- 0.13)	93.27 (+ 1.17)	90.11 (+ 1.46)	93.7 (+ 0.5)

4.2 3D Dense Prediction Task

For dense prediction tasks, including part and semantic segmentation, we adopt the standard task-specific heads of the corresponding baselines for fair comparison. To further establish the practical efficiency of Mantis, we additionally report broader resource metrics in Appendix B.6.

Part Segmentation. As shown in Table 2a, in this challenging fine-grained understanding task(Yi et al., 2016), our approach still achieves competitive results on instance-level mIoU (Inst. mIoU) and consistent improvements on class-level mIoU (Cls. mIoU) compared with full fine-tuning. Qualitatively, as shown in Figure 6 (b), Mantis produces more accurate and coherent part boundaries than full fine-tuning, especially for small or structurally complex parts, thereby preserving fine-grained geometric structures.

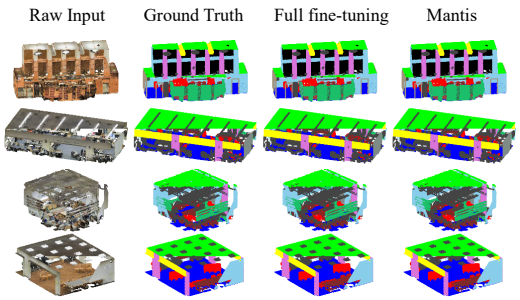


Figure 4: Qualitative comparison on the S3DIS.

Semantic Segmentation. As shown in Table 2b, we report mean class accuracy (mAcc) and mean IoU (mIoU) on S3DIS Area 5 (Armeni et al., 2016), and voxel-level mIoU (VmIoU) and point-level mIoU (PmIoU) on ScanNet V2 (Dai et al., 2017). Due to their pretraining primarily on object-level datasets, these backbones struggle to transfer effectively to large-scale indoor scene segmentation(Liang et al., 2025). In contrast, Mantis adapts state dynamics while reducing serialization variation, bridging the gap between fully and efficiently fine-tuning strategies. With Mamba3D as the backbone on S3DIS, Mantis achieves 68.0% mAcc and 58.1% mIoU with only 0.7M trainable backbone parameters,

Table 2: Results on large-scale semantic segmentation on S3DIS (Armeni et al., 2016) and ScanNet V2 (Dai et al., 2017), and part segmentation on ShapeNetPart (Yi et al., 2016). Only trainable backbone parameters (Param.) are reported, as the increase mainly comes from the shared task head.

(a) Part segmentation results.

Methods	Param.	Inst.mIoU	Cls.mIoU
Self-Supervised Learning (Full Fine-tuning)			
Point-BERT (Yu et al., 2022)	21.9	85.6	84.1
Point-MAE (Pang et al., 2022)	21.9	86.1	84.1
ACT(Dong et al., 2023)	27.1	86.1	**84.7**
ReCon (Qi et al., 2023)	43.3	**86.4**	84.5
Self-Supervised Learning (Parameter-Efficient Fine-Tuning)			
PointMamba	12.2	85.3	82.6
+ PMA	1.3	84.2 (- 1.1)	82.3 (- 0.3)
+ Mantis	0.6	85.1 (- 0.2)	82.9 (+ 0.3)
Mamba3D	15.5	85.6	83.6
+ PMA	1.8	83.7 (- 1.9)	82.4 (- 1.2)
+ Mantis	0.7	86.1 (+ 0.5)	84.1 (+ 0.5)
ZigzagPointMamba	12.2	85.8	84.2
+ PMA	1.3	84.1 (- 1.7)	82.0 (- 2.2)
+ Mantis	0.6	85.9 (+ 0.1)	84.5 (+ 0.3)

(b) Indoor scene semantic segmentation results.

Method	Param.	S3DIS Area 5		ScanNet V2	
		mAcc	mIoU	VmIoU	PmIoU
Self-Supervised Learning (Full Fine-tuning)					
Point-BERT (Yu et al., 2022)	21.8	69.7	60.5	49.9	49.6
ReCon (Qi et al., 2023)	21.8	69.3	60.4	50.8	50.5
Self-Supervised Learning (Parameter-Efficient Fine-Tuning)					
Point-MAE	21.8	69.9	60.8	51.2	50.8
+ PointLoRA	2.2	64.8	52.6	38.1	37.2
+ PMA	2.4	65.0	53.2	39.3	37.9
+ PoinTriE	1.9	66.1	54.9	43.4	42.1
PointMamba	12.2	70.1	60.9	50.3	50.1
+ PMA	1.2	63.8	54.6	42.7	41.6
+ Mantis	0.6	69.3	59.8	49.8	49.5
Mamba3D	15.4	68.2	58.4	51.3	50.7
+ PMA	1.8	64.1	54.6	43.1	42.3
+ Mantis	0.7	68.0	58.1	50.4	50.0

Table 3: Ablation study on the setting of Mantis, including different components, inserted layers and PEFT methods. The details of the table are described in Sec. 4.3.

(a) The effect of each component.

SAA	DSCD $\mathcal{L}_{\text{feat}}$	DSCD $\mathcal{L}_{\text{pred}}$	#TP (M)	PB_T50_RS
Vanilla Protocol				
Full fine-tuning			16.9	92.05
Linear Probing			0.27	80.84
	✓		0.38	84.12
		✓	0.38	83.95
	✓	✓	0.38	84.53
✓			0.72	91.86
✓	✓		0.84	92.61
✓		✓	0.84	92.38
✓	✓	✓	0.84	**93.48**

(b) The effect of inserted layers.

Layers	#TP (M)	PB_T50_RS
	0.27	80.84
1→3	0.50	88.74
1→6	0.61	90.12
1→9	0.72	91.56
1→12	0.84	**93.48**
4→6	0.50	89.21
4→9	0.61	90.37
4→12	0.72	92.04
7→9	0.50	88.63
7→12	0.61	89.72
10→12	0.50	88.11

(c) The effect of different PEFT methods.

Methods	#TP (M)	PB_T50_RS
Mamba3D	16.90	92.05
+ Adapter(Houlsby et al., 2019)	0.87	84.45 (- 7.60)
+ Prefix tuning(Li & Liang, 2021)	0.71	83.26 (- 8.79)
+ LoRA(Hu et al., 2022)	0.91	83.19 (- 8.86)
+ VPT-Deep(Jia et al., 2022)	0.41	83.37 (- 8.68)
+ AdaptFormer(Chen et al., 2022)	0.87	84.13 (- 7.92)
+ BI-AdaptFormer(Jie et al., 2023)	0.87	82.88 (- 9.17)
+ ProDiaL(Ham et al., 2025)	0.97	83.81 (- 8.24)
+ PointLoRA(Wang et al., 2025b)	0.77	85.36 (- 6.69)
+ PointGST(Liang et al., 2025)	0.62	87.38 (- 4.67)
+ Mantis	0.84	**93.48** (+ 1.43)

closely matching full fine-tuning at 68.2% and 58.4%. As further shown in Figure 4 and Figure 5, Mantis produces reliable dense predictions in complex indoor scenes. Overall, Mantis remains competitive with full fine-tuning on these challenging tasks, an uncommon achievement in PEFT.

4.3 Ablation Study

We conduct ablation studies on the most challenging PB_T50_RS variant based on Mamba3D to investigate the rationale and effectiveness of Mantis. More results are provided in Appendix B.2.

Ablation on different components. We first study the contribution of each component in Mantis. Table 3a shows that SAA is the primary source of performance improvement. DSCD alone provides modest improvement, whereas introducing SAA leads to substantial gains. Building on SAA, both feature consistency and prediction consistency bring further gains, and their combination achieves the best result. Overall, DSCD functions as a regularizer that complements SAA, while feature- and prediction-level consistency provide complementary benefits.

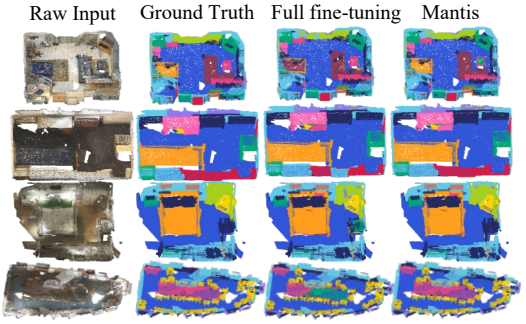


Figure 5: Qualitative results on the ScanNet V2.

Ablation on different inserted layers. One straightforward way to further reduce the number of tunable parameters is to insert SAA into only a subset of encoder layers rather than the full network. However, as illustrated in Table 3b, restricting SAA to partial layers consistently degrades performance, regardless of whether the selected layers are shallow, intermediate, or deep. This observation suggests that effective adaptation of a frozen Mamba backbone requires state modulation

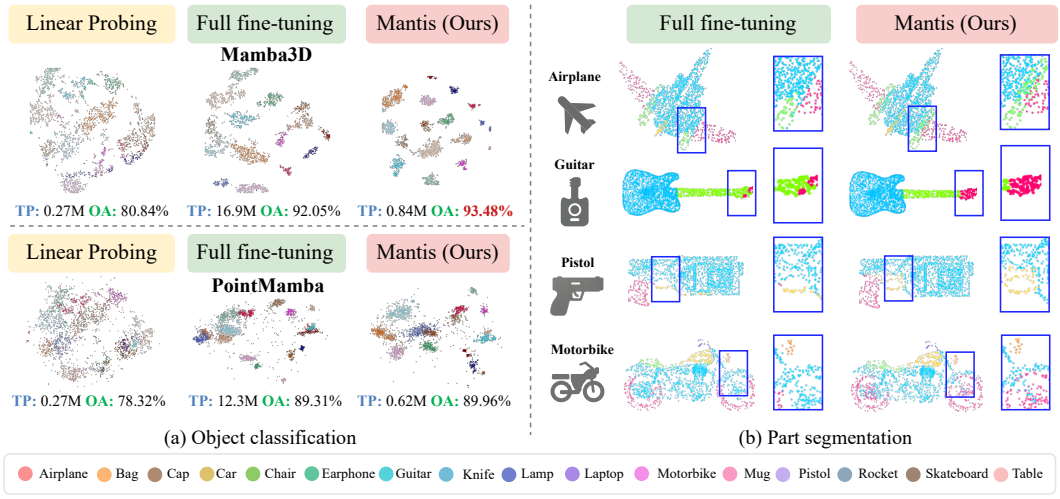


Figure 6: **Qualitative analysis results for object classification and part segmentation.** (a) The t-SNE visualization results on the PB_T50_RS variant with different fine-tuning schemes. (b) Comparison of full fine-tuning and Mantis on part segmentation with Mamba3D(Han et al., 2024).

across all encoder layers, rather than within a few isolated layers. When SAA is limited to middle or late layers, the performance drops noticeably, indicating that suboptimal state evolution formed in earlier stages cannot be adequately compensated by later-layer correction.

Ablation on different PEFT methods. To further validate the effectiveness of Mantis, we compare Mantis with several representative PEFT approaches originally developed for NLP(Houlsby et al., 2019; Li & Liang, 2021; Hu et al., 2022), 2D vision(Jia et al., 2022; Chen et al., 2022; Jie et al., 2023; Ham et al., 2025), and 3D point clouds(Liang et al., 2025; Wang et al., 2025b). As illustrated in Table 3c, although these methods introduce different forms of parameter-efficient adaptation, they all suffer from substantial performance degradation compared with full fine-tuning. This suggests that existing PEFT strategies, even those tailored for point clouds, remain insufficient to effectively adapt frozen Mamba backbones. A possible reason is that most of them are designed for token-wise interactions or feature transformation, and therefore cannot properly align with the state-space dynamics and serialization-sensitive modeling mechanism of Mamba. In contrast, the strength of Mantis lies not merely in parameter efficiency, but in its Mamba-native design that better matches the intrinsic adaptation demands of selective SSMs.

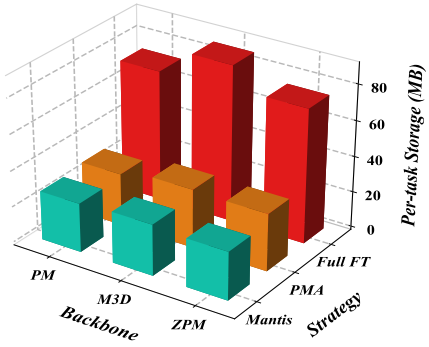


Figure 7: Per-task storage comparison.

Analysis on storage efficiency. Since task-specific parameters must be independently maintained, per-task storage constitutes a critical scalability factor when adapting a shared pretrained backbone to multiple downstream tasks. As shown in Figure 7, Mantis consistently incurs lower storage overhead across all three Mamba backbones. Full fine-tuning requires *2.66×, 3.03×, and 2.76×* the per-task storage of Mantis on PointMamba, Mamba3D, and ZigzagPointMamba, respectively. Moreover, Mantis further reduces storage by 5.8%, 12.7%, and 15.8% compared with PMA(Zha et al., 2025a), demonstrating more favorable storage scalability for parameter-efficient downstream adaptation.

5 Conclusion

In this paper, we propose Mantis, the first Mamba-native PEFT framework for 3D PFMs. We identify the mismatch between existing Transformer-native PEFT methods and Mamba-based backbones, and address it through state-level adaptation and cross-serialization regularization. Extensive experiments demonstrate that Mantis achieves strong downstream performance while training less than 5% of the backbone parameters. As an early exploration of Mamba-native PEFT, we expect Mantis can serve as a baseline and provide insights for future 3D PEFT research. Building on this foundation, extending Mamba-native PEFT to complementary modalities remains a promising but underexplored direction.

6 STATEMENTS

AI USE STATEMENT

Generative AI tools were used only to assist with language editing and presentation refinement. All technical content, experimental results, analyses, and conclusions were produced and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work is based on publicly available 3D point cloud benchmarks and does not involve human-subject studies or the collection of private or personally identifiable information. We are not aware of any additional ethical concerns arising from this research.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training settings, evaluation protocols, and implementation details in the main paper and appendix. Experiments follow the standard dataset splits and evaluation settings of the corresponding benchmarks, and our reported results are averaged over three random seeds where applicable. The source code and configuration files are publicly available to ensure the reproducibility of our results.

REFERENCES

Zixiang Ai, Zichen Liu, Yuanhang Lei, Zhenyu Cui, Xu Zou, and Jiahuan Zhou. GAPrompt: Geometry-aware point cloud prompt for 3D vision model. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Ali Bahri, Moslem Yazdanpanah, Mehrdad Noori, Sahar Dastani, Milad Cheraghalikhani, Gustavo Adolfo Vargas Hakim, David Osowiechi, Farzad Beizaee, Ismail Ben Ayed, and Christian Desrosiers. Spectral informed mamba for robust point cloud processing. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (NAACL)*, 2019.

Linshuang Diao, Sensen Song, Yurong Qian, and Dayong Ren. Zigzagpointmamba: Spatial-semantic mamba for point cloud understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling (COLM)*, 2024.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

Ziyu Guo, Renrui Zhang, Longtian Qiu, Xianzhi Li, and Pheng-Ann Heng. Joint-mae: 2d-3d joint masked autoencoders for 3d point cloud pre-training. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

Seokil Ham, Hee-Seon Kim, Sangmin Woo, and Changick Kim. Parameter efficient mamba tuning via projector-targeted diagonal-centric linear transformation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30106–30115. IEEE, 2025.

Xu Han, Yuan Tang, Zhaoxuan Wang, and Xianzhi Li. Mamba3d: Enhancing local features for 3d point cloud analysis via state space model. In *Proceedings of the 32nd ACM International Conference on Multimedia (ACM MM)*, 2024.

Xu Han, Yuan Tang, Jinfeng Xu, and Xianzhi Li. Most: Efficient monarch sparse tuning for 3d representation learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*. Springer, 1935.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning (ICML)*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision (ECCV)*. Springer, 2022.

Zhaoyi Jiang, Yi Xu, Frederick W.B. Li, Gary K.L. Tam, Chao Song, and Bailin Yang. Octmamba: Mamba-based octree context entropy model for point cloud geometry compression. *Pattern Recognition (PR)*, 2026.

Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2023.

Pengbo Li, Yiding Sun, and Haozhe Cheng. Pointdico: Contrastive 3d representation learning guided by diffusion models. In *2025 International Joint Conference on Neural Networks (IJCNN)*, 2025.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (ACL)*, 2021.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems (NeurIPS)*, 2018.

Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Dingkang Liang, Tianrui Feng, Xin Zhou, Yumeng Zhang, Zhikang Zou, and Xiang Bai. Parameter-efficient fine-tuning in spectral domain for point cloud learning. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 2025.

Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision (ECCV)*. Springer, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2019.

Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yijun Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European Conference on Computer Vision (ECCV)*, 2022.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017b.

Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Kanglin Qu, Pan Gao, Qun Dai, and Yuanhao Sun. Hydramamba: Multi-head state space model for global point cloud learning. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, 2025.

Kanglin Qu, Pan Gao, Qun Dai, Zhanzhi Ye, Rui Ye, and Yuanhao Sun. Cloudmamba: Grouped selective state spaces for point cloud analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Yiding Sun, Haozhe Cheng, Chaoyi Lu, Zhengqiao Li, Minghong Wu, Huimin Lu, and Jihua Zhu. Hyperpoint: Multimodal 3d foundation model in hyperbolic space. *Pattern Recognition (PR)*, 2026a.

Yiding Sun, Chaoyi Lu, Haozhe Cheng, Jun Wang, Huimin Lu, Lin Chen, and Jihua Zhu. Curve3d: Curvature-aware masked autoencoder for self-supervised point cloud understanding. *Pattern Recognition (PR)*, 2026b.

Yiding Sun, Dongxu Zhang, Jihua Zhu, Haozhe Cheng, Zhengqiao Li, Pengcheng Li, Chaowei Fang, Yonghao Dong, and Lin Chen. Tri-efficient transfer learning for point cloud videos. In *European Conference on Computer Vision (ECCV)*, 2026c.

Yiding Sun, Jihua Zhu, Haozhe Cheng, Chaoyi Luv, Zhichuan Yang, Lin Chen, and Yaonan Wang. Align then adapt: Rethinking parameter-efficient transfer learning in 4d perception. *IEEE Transactions on Multimedia (TMM)*, 2026d.

Yiwen Tang, Ray Zhang, Zoey Guo, Xianzheng Ma, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Point-peft: Parameter-efficient fine-tuning for 3d pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, 2017.

Chuxin Wang, Yixin Zha, Wenfei Yang, and Tianzhu Zhang. Strumamba3d: Exploring structural mamba for self-supervised point cloud representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025a.

Song Wang, Xiaolu Liu, Lingdong Kong, Jianyun Xu, Chunyong Hu, Gongfan Fang, Wentong Li, Jianke Zhu, and Xinchao Wang. Pointlora: Low-rank adaptation with token selection for point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Yankai Wang, Yiding Sun, Qirui Wang, Pengbo Li, Chaoyi Lu, and Dongxu Zhang. Pointrft: Explicit reinforcement fine-tuning for point cloud few-shot learning. *arXiv preprint arXiv:2603.23957*, 2026.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *CVPR*, 2025.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 2016.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Yaohua Zha, Jinpeng Wang, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. Instance-aware dynamic prompt tuning for pre-trained point cloud models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Yaohua Zha, Huizhen Ji, Jinmin Li, Rongsheng Li, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. Towards compact 3d representations via point feature enhancement masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Yaohua Zha, Yanzi Wang, Hang Guo, Jinpeng Wang, Tao Dai, Bin Chen, Zhihao Ouyang, Xue Yuerong, Ke Chen, and Shu-Tao Xia. Pma: Towards parameter-efficient point cloud understanding via point mamba adapter. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2025a.

Yixin Zha, Chuxin Wang, Wenfei Yang, Tianzhu Zhang, and Feng Wu. Exploring vision semantic prompt for efficient point cloud understanding. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025b.

Dongxu Zhang, Yiding Sun, Pengcheng Li, Yumou Liu, Hongqiang Lin, Haoran Xu, Xiaoxuan Mu, Liang Lin, Wenbiao Yan, Ning Yang, et al. Pointcot: A multi-modal benchmark for explicit 3d geometric reasoning. *arXiv preprint arXiv:2602.23945*, 2026.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Renrui Zhang, Liuhui Wang, Yu Qiao, Peng Gao, and Hongsheng Li. Learning 3d representations from 2d pre-trained models via image-to-point masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2023.

Tao Zhang, Haobo Yuan, Lu Qi, Jiangning Zhang, Qianyu Zhou, Shunping Ji, Shuicheng Yan, and Xiangtai Li. Point cloud mamba: Point cloud learning via state space model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Xin Zhou, Dingkang Liang, Wei Xu, Xingkui Zhu, Yihan Xu, Zhikang Zou, and Xiang Bai. Dynamic adapter meets prompt tuning: Parameter-efficient transfer learning for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

APPENDIX TABLE OF CONTENTS

A Additional theoretical analysis 16

A.1 Preliminaries: State Space Models and PointMamba 16

A.2 A closer look at selective SSM under serialization 16

A.3 Effect of State-Aware Adapter on selective state-space dynamics 17

A.4 Parameter and complexity analysis 18

B More experimental results 19

B.1 Training detail 19

B.2 Additional ablation study 20

B.3 Few-shot Learning 21

B.4 Detailed results for classification and part segmentation 21

B.5 Part segmentation visualization 22

B.6 Additional Results on Large-scale Semantic Segmentation 22

C Explanatory experiments and discussions 23

C.1 Training dynamics and optimization stability 23

C.2 Further analysis on PB_T50_RS 24

C.3 Discussion on the scope of Mamba-native adaptation 25

D Symbol table 26

A Additional theoretical analysis

A.1 Preliminaries: State Space Models and PointMamba

Drawing inspiration from control theory, State Space Models (SSMs) map an input sequence x_t to an output y_t through a latent state h_t. In its discretized form, the transformation can be written as:

$$h_t = \overline{\boldsymbol{A}} h_{t-1} + \overline{\boldsymbol{B}} x_t, \qquad y_t = \boldsymbol{C} h_t, \tag{12}$$

where $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{B}}$ are the discretized state transition and input projection matrices. While early structured SSMs rely on fixed transition parameters, Mamba (S6)(Gu & Dao, 2024) introduces input-dependent parameters $(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Delta})$, enabling selective and content-aware sequence modeling.

For point cloud analysis, PointMamba(Liang et al., 2024) adopts the SSM framework by treating serialized point tokens as sequential inputs. The state update can be written as:

$$h_t = \overline{\boldsymbol{A}}_t h_{t-1} + \overline{\boldsymbol{B}}_t x_t, \qquad y_t = \boldsymbol{C}_t h_t, \tag{13}$$

where h_t denotes the hidden state and x_t denotes the input point token. Mamba keeps the structured continuous-time transition $\boldsymbol{A}$ fixed, while $\boldsymbol{B}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{\Delta}_t$ are input-dependent. Accordingly, the discretized transition $\overline{\boldsymbol{A}}_t = \exp(\boldsymbol{A}\boldsymbol{\Delta}_t)$ remains input-dependent through $\boldsymbol{\Delta}_t$. This selective state update provides efficient sequence modeling, but also causes the hidden-state propagation to depend on the serialized point order.

A.2 A closer look at selective SSM under serialization

Different from LTI SSMs with fixed transition kernels, Mamba keeps the structured continuous-time transition $\boldsymbol{A}$ fixed while using input-dependent selective coefficients $\boldsymbol{B}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{\Delta}_t$. Consequently, the effective discretized transition $\overline{\boldsymbol{A}}_t = \exp(\boldsymbol{A}\boldsymbol{\Delta}_t)$ also becomes input-dependent through $\boldsymbol{\Delta}_t$. For point cloud modeling, different valid serializations therefore induce different input sequences and selective operator chains, making the effective transfer kernel dependent on the chosen serialization.

As described in Sec. A.1, for branch $k \in \{1, 2\}$, the selective state-space model updates the hidden state according to

$$h_t^{(k,0)} = \overline{\boldsymbol{A}}_t^{(k)} h_{t-1}^{(k,0)} + \overline{\boldsymbol{B}}_t^{(k)} x_t^{(k)}, \qquad y_t^{(k,0)} = \boldsymbol{C}_t^{(k)} h_t^{(k,0)}, \tag{14}$$

where $x_t^{(k)} \in \mathbb{R}^d$ denotes the serialized input token, $\overline{\boldsymbol{A}}_t^{(k)}$ and $\overline{\boldsymbol{B}}_t^{(k)}$ are the discretized operators of the frozen selective SSM, and $\boldsymbol{C}_t^{(k)}$ is the corresponding input-dependent readout coefficient.

Ignoring the initial state for simplicity, the recurrent form in Eq. equation 14 can be expanded as:

$$h_t^{(k,0)} = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(k)} \right) \overline{\boldsymbol{B}}_i^{(k)} x_i^{(k)}, \tag{15}$$

where the product is defined in temporal order, *i.e.*, $\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(k)} = \overline{\boldsymbol{A}}_t^{(k)} \overline{\boldsymbol{A}}_{t-1}^{(k)} \cdots \overline{\boldsymbol{A}}_{i+1}^{(k)}$, and equals the identity matrix when $i = t$.

Substituting Eq. equation 15 into the readout equation yields:

$$y_t^{(k,0)} = \sum_{i=1}^{t} \boldsymbol{C}_t^{(k)} \left(\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(k)} \right) \overline{\boldsymbol{B}}_i^{(k)} x_i^{(k)}. \tag{16}$$

Thus, the selective transfer kernel from step i to step t is defined as:

$$\boldsymbol{W}_{t,i}^{(k,0)} = \boldsymbol{C}_t^{(k)} \left(\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(k)} \right) \overline{\boldsymbol{B}}_i^{(k)}, \qquad 1 \le i \le t \le n, \tag{17}$$

where n is the serialized sequence length. Then Eq. equation 16 can be rewritten as:

$$y_t^{(k,0)} = \sum_{i=1}^{t} \boldsymbol{W}_{t,i}^{(k,0)} x_i^{(k)}. \tag{18}$$

Stacking all outputs $\hat{\boldsymbol{y}}^{(k,0)} = [y_1^{(k,0)}, \ldots, y_n^{(k,0)}]^\top$ and inputs $\hat{\boldsymbol{x}}^{(k)} = [x_1^{(k)}, \ldots, x_n^{(k)}]^\top$, Eq. equation 18 can be expressed in matrix form:

$$\hat{\boldsymbol{y}}^{(k,0)} = \boldsymbol{W}^{(k,0)} \hat{\boldsymbol{x}}^{(k)}, \qquad \boldsymbol{W}^{(k,0)} = \begin{bmatrix} \boldsymbol{W}_{1,1}^{(k,0)} & 0 & \cdots & 0 \\ \boldsymbol{W}_{2,1}^{(k,0)} & \boldsymbol{W}_{2,2}^{(k,0)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{W}_{n,1}^{(k,0)} & \boldsymbol{W}_{n,2}^{(k,0)} & \cdots & \boldsymbol{W}_{n,n}^{(k,0)} \end{bmatrix}. \tag{19}$$

Since $\boldsymbol{W}^{(k,0)}$ is lower triangular, the selective SSM defines a causal, input-dependent transfer matrix over the serialized point sequence.

Eq. equation 19 also reveals the source of serialization sensitivity. For two valid serializations of the same point cloud, not only the input order $\hat{\boldsymbol{x}}^{(1)}$ and $\hat{\boldsymbol{x}}^{(2)}$ differ, but the induced operator chains $\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(1)}$ and $\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(2)}$ also differ. Consequently, the same underlying geometry may lead to different transfer matrices and hence different representations under distinct valid serializations.

A.3 Effect of State-Aware Adapter on selective state-space dynamics

To characterize the state-level adaptation mechanism of SAA, we analyze its effect on the selective transfer kernel of the frozen Mamba backbone. Building on the selective transfer kernel derived in Appendix A.2, $\boldsymbol{W}_{t,i}^{(k)}$ denotes the selective transfer kernel after SAA modulation, while $\boldsymbol{W}_{t,i}^{(k,0)}$ denotes its frozen counterpart without SAA:

$$\boldsymbol{W}_{t,i}^{(k,0)} = \boldsymbol{C}_t^{(k)} \left(\prod_{j=i+1}^{t} \overline{\boldsymbol{A}}_j^{(k)} \right) \overline{\boldsymbol{B}}_i^{(k)}. \tag{20}$$

where $\overline{\boldsymbol{A}}_t^{(k)}$ and $\overline{\boldsymbol{B}}_t^{(k)}$ are the discretized operators of the original frozen selective SSM, and $\boldsymbol{C}_t^{(k)}$ is the corresponding readout matrix. Under SAA, we first define the coefficient-specific factorized correction as:

$$\boldsymbol{\Gamma}_{O,t}^{(k)} = \mathbf{U}_O \operatorname{diag}\big(\mathbf{u}_t^{(k)}\big) \mathbf{V}_O, \qquad O \in \{B, C, \Delta\}, \tag{21}$$

where $\mathbf{u}_t^{(k)} \in \mathbb{R}^r$ is the shared sparse control signal, and $\mathbf{U}_O$ and $\mathbf{V}_O$ are coefficient-specific low-rank modulation matrices. The actual native-layout correction is then given by

$$\delta_{O,t}^{(k)} = \mathcal{R}_O\left(\boldsymbol{\Gamma}_{O,t}^{(k)}\right), \qquad \widetilde{\boldsymbol{O}}_t^{(k)} = \boldsymbol{O}_t^{(k)} + \delta_{O,t}^{(k)}. \tag{22}$$

Here, $\mathcal{R}_O(\cdot)$ maps the factorized correction to the native coefficient layout. The structured continuous-time transition $\boldsymbol{A}$ remains frozen. For completeness, in the standard Mamba selective scan, $\boldsymbol{B}_t$ and $\boldsymbol{C}_t$ are token-wise state coefficients in $\mathbb{R}^{d_s}$, while $\boldsymbol{\Delta}_t \in \mathbb{R}^{d_{\text{in}}}$ is defined over the inner channels, where d_s and d_{in} denote the SSM state size and Mamba inner dimension, respectively. Accordingly, $\mathcal{R}_O(\cdot)$ is implemented as a parameter-free reshape/permutation that converts the factorized correction to the native tensor layout of coefficient O before residual addition.

$$\boldsymbol{W}_{t,i}^{(k)} = \widetilde{\boldsymbol{C}}_t^{(k)} \left(\prod_{j=i+1}^{t} \widehat{\boldsymbol{A}}_j^{(k)} \right) \widehat{\boldsymbol{B}}_i^{(k)}. \tag{23}$$

The corresponding kernel perturbation is

$$\Delta \boldsymbol{W}_{t,i}^{(k)} = \boldsymbol{W}_{t,i}^{(k)} - \boldsymbol{W}_{t,i}^{(k,0)}. \tag{24}$$

For each selective coefficient O, branch k, and sequence position t, the factorized correction $\mathbf{\Gamma}_{O,t}^{(k)}$ is a sum of at most $|\mathbf{u}_t^{(k)}|_0$ active rank-one components. Therefore,

$$\operatorname{rank}\left(\mathbf{\Gamma}_{O,t}^{(k)}\right) \le |\mathbf{u}_t^{(k)}|_0 \le r. \tag{25}$$

Therefore, SAA constructs each native coefficient correction from a sparse low-rank factorized source. The low-rank property applies to $\mathbf{\Gamma}_{O,t}^{(k)}$, while $\mathcal{R}_O(\cdot)$ only maps this structured correction to the native coefficient layout. Through discretization and recurrent propagation, these native-layout corrections induce structured changes in the selective transfer kernel and pretrained state dynamics. This factorized parameterization restricts trainable adaptation to a low-dimensional subspace and explains the parameter efficiency of SAA.

We further examine how the resulting native-layout coefficient corrections affect the hidden-state trajectory through discretization and recurrence. The hidden states produced by the frozen backbone and the SAA-modulated backbone are denoted as $h_t^{(k,0)}$ and $h_t^{(k)}$, respectively:

$$h_t^{(k,0)} = \overline{\boldsymbol{A}}_t^{(k)} h_{t-1}^{(k,0)} + \overline{\boldsymbol{B}}_t^{(k)} x_t^{(k)}, \qquad h_t^{(k)} = \widehat{\boldsymbol{A}}_t^{(k)} h_{t-1}^{(k)} + \widehat{\boldsymbol{B}}_t^{(k)} x_t^{(k)}. \tag{26}$$

The hidden-state deviation $\delta h_t^{(k)} = h_t^{(k)} - h_t^{(k,0)}$ satisfies

$$\begin{aligned} \delta h_t^{(k)} = \overline{\boldsymbol{A}}_t^{(k)} \delta h_{t-1}^{(k)} &+ \left(\widehat{\boldsymbol{A}}_t^{(k)} - \overline{\boldsymbol{A}}_t^{(k)}\right) h_{t-1}^{(k)} \\ &+ \left(\widehat{\boldsymbol{B}}_t^{(k)} - \overline{\boldsymbol{B}}_t^{(k)}\right) x_t^{(k)}. \end{aligned} \tag{27}$$

Thus, the recurrent deviation is propagated by the frozen transition $\overline{\boldsymbol{A}}_t^{(k)}$, while SAA contributes an additive transition perturbation. Assume that $\|\overline{\boldsymbol{A}}_t^{(k)}\|_2 \le \rho < 1$, $\|h_t^{(k)}\|_2 \le H$, and $\|x_t^{(k)}\|_2 \le X$. Since $\boldsymbol{A}$ remains frozen and $\widehat{\boldsymbol{A}}_t^{(k)} = \exp(\boldsymbol{A}\widetilde{\boldsymbol{\Delta}}_t^{(k)})$, a bounded correction $\|\widetilde{\boldsymbol{\Delta}}_t^{(k)} - \boldsymbol{\Delta}_t^{(k)}\|_2 \le \varepsilon_\Delta$ induces, over a bounded discretization domain,

$$\left\|\widehat{\boldsymbol{A}}_t^{(k)} - \overline{\boldsymbol{A}}_t^{(k)}\right\|_2 \le \varepsilon_{\mathrm{tr}}, \qquad \left\|\widehat{\boldsymbol{B}}_t^{(k)} - \overline{\boldsymbol{B}}_t^{(k)}\right\|_2 \le \varepsilon_B, \tag{28}$$

where $\varepsilon_{\mathrm{tr}} = O(\varepsilon_\Delta)$ follows from the local Lipschitz continuity of the matrix exponential.

The hidden-state deviation is therefore bounded by

$$\left\|\delta h_t^{(k)}\right\|_2 \le \rho \left\|\delta h_{t-1}^{(k)}\right\|_2 + \varepsilon_{\mathrm{tr}} H + \varepsilon_B X, \tag{29}$$

which yields

$$\left\|\delta h_t^{(k)}\right\|_2 \le \rho^t \left\|\delta h_0^{(k)}\right\|_2 + \frac{1-\rho^t}{1-\rho}\left(\varepsilon_{\mathrm{tr}} H + \varepsilon_B X\right). \tag{30}$$

The modulation of $\boldsymbol{C}_t$ affects the output rather than the hidden-state recurrence. If $\|\boldsymbol{C}_t^{(k)}\|_2 \le M_C$ and $\|\widetilde{\boldsymbol{C}}_t^{(k)} - \boldsymbol{C}_t^{(k)}\|_2 \le \varepsilon_C$, then

$$\left\|y_t^{(k)} - y_t^{(k,0)}\right\|_2 \le M_C \left\|\delta h_t^{(k)}\right\|_2 + \varepsilon_C H. \tag{31}$$

Therefore, under the stated boundedness assumptions, SAA induces bounded perturbations in both the hidden-state trajectory and the selective-SSM output. This provides a conditional perturbation analysis of the adapted state dynamics, while the optimization stability observed in practice is supported separately by the empirical training dynamics.

A.4 Parameter and complexity analysis

We finally analyze the additional parameter and computational overhead introduced by the proposed State-Aware Adapter (SAA). Recall that SAA consists of the pre-scan control projections W_x^ϕ, W_z^ϕ, and W_e^ϕ, the control-feature fusion process $\Phi(\cdot)$, the driving and gating projections W_{drv}^ψ and W_{gt}^ψ,

and the coefficient-specific low-rank modulation matrices $\mathbf{U}_O$ and $\mathbf{V}_O$. Accordingly, the total number of trainable parameters in one SAA module is

$$|\Theta_{\mathrm{SAA}}| = |W_x^\phi| + |W_z^\phi| + |W_e^\phi| + |\Theta_\Phi| + |W_{\mathrm{drv}}^\psi| + |W_{\mathrm{gt}}^\psi| + \sum_{O\in\{B,C,\Delta\}} \left(|\mathbf{U}_O| + |\mathbf{V}_O|\right). \quad (32)$$

Specifically, $W_x^\phi, W_z^\phi, W_e^\phi \in \mathbb{R}^{d_\phi \times d}$. and $W_{\mathrm{drv}}^\psi, W_{\mathrm{gt}}^\psi \in \mathbb{R}^{r\times d_\phi}$. In our implementation, the control-feature fusion process $\Phi(\cdot)$ is instantiated as a lightweight MLP that projects the concatenated $3d_\phi$-dimensional control feature into the shared d_ϕ-dimensional control space. For each selective coefficient $O \in \{B, C, \Delta\}$, we denote $\mathbf{U}_O \in \mathbb{R}^{d_O^{\mathrm{out}}\times r}$ and $\mathbf{V}_O \in \mathbb{R}^{r\times d_O^{\mathrm{in}}}$, where d_O^{out} and d_O^{in} follow the native dimensions of selective coefficient O, and r is the bottleneck dimension. The modulation preserves the native layout of each selective coefficient without introducing dense transition matrices. The total number of trainable parameters in one SAA module therefore becomes

$$|\Theta_{\mathrm{SAA}}| = 3d_\phi d + 3d_\phi^2 + d_\phi + 2rd_\phi + \sum_{O\in\{B,C,\Delta\}} r\left(d_O^{\mathrm{out}} + d_O^{\mathrm{in}}\right). \quad (33)$$

Eq. equation 33 shows that the adaptation overhead is dominated by low-dimensional control projections and low-rank modulation. Since $r \ll d$ and d_ϕ is small, the number of trainable parameters introduced by SAA remains much smaller than that of full fine-tuning.

We next analyze the computational overhead. SAA introduces only token-wise control-feature projection, control-feature fusion, sparse control generation, and factorized coefficient modulation. All these operations are computed independently across sequence positions, so the additional overhead scales linearly with the sequence length n. More importantly, all SAA coefficients are generated in parallel before recurrence and consumed by a single native selective-scan call per SAA-Mamba block and serialization branch. Hence, SAA introduces no additional recurrent pass and preserves the native linear-time selective-scan structure.

B More experimental results

B.1 Training detail

Table 4: Implementation details for various downstream tasks.

Configuration	Classification		Segmentation		
	ModelNet40	ScanObjectNN	ShapeNetPart	S3DIS Area5	ScanNet
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	3e-4	5e-4	2e-4	1e-4	1e-3
Weight decay	5e-2	5e-2	5e-2	5e-2	5e-2
Learning rate scheduler	cosine	cosine	cosine	cosine	cosine
Training epochs	300	300	300	100	200
Warm-up epochs	10	10	10	5	10
Batch size	32	32	16	4	4
r in State-Aware Adapter	8	8	8	8	8
Softmax temperature τ	1	1	1	1	1
Num. of encoder layers	12	12	12	12	12
Input points	1024	2048	2048	24576	32768
Num. of patches	64	128	128	192	256
Patch size	32	32	32	48	48
Augmentation	Scale&Trans/Rotation	Scale&Trans	Scale&Trans	Scale&Trans/Rotation	Scale&Trans/Rotation

All experiments, including classification, few-shot, and segmentation, follow the standard dataset splits as defined in the respective benchmarks, and results are averaged over multiple runs with different random seeds to ensure reproducibility. For comparison, full fine-tuning retains the native backbone architecture, whereas the lightweight order-conditioning and dual-serialization components are specific to Mantis and are fully included in its reported parameter and computational budgets.

We adopt downstream fine-tuning configurations in alignment with the pioneering work Point-Mamba(Liang et al., 2024), as in Table 4. For example, when fine-tuning on ModelNet40(Wu et al., 2015), the training process spans 300 epochs, using a cosine learning rate scheduler that starts at 3e-4, with a 10-epoch warm-up period. The AdamW optimizer is employed. All experiments are conducted on a single GeForce RTX 5090 using PyTorch version 2.7.1+cu128.

Table 5: Ablation study on Mantis based on Mamba3D(Han et al., 2024), including sparse controller design, bottleneck dimension r, and scanning curves.

(a) The effect of different scanning curves.

Scanning curve	PB_T50_RS (%)
Random	91.74
Hilbert and Trans-Hilbert	**93.48**
Z-order and Trans-Z-order	92.86
Hilbert and Z-order	93.02
Trans-Hilbert and Trans-Z-order	<u>93.21</u>

(b) The effect of r.

r	#TP (M)	PB_T50_RS (%)
8	0.84	93.48
16	1.00	<u>93.50</u>
32	1.32	**93.58**
64	1.95	93.47
72	2.12	93.41

(c) The effect of controller design.

Controller	#TP (M)	PB_T50_RS (%)
Dense-MLP	0.95	92.71
Sigmoid Gate	0.84	92.88
Tanh Gate	0.84	92.97
Hard-threshold	0.84	<u>93.16</u>
Soft-threshold	0.84	**93.48**

Table 6: Ablation study on the internal design of SAA based on Mamba3D(Han et al., 2024), including state fusion function and modulated state-space operators.

(a) Ablation on fusion function $\Phi(\cdot)$.

Fusion function $\Phi(\cdot)$	#TP (M)	PB_T50_RS (%)
Addition	0.81	92.34
Concatenation	0.82	92.71
Gated Fusion	0.86	<u>93.21</u>
Cross-attention Fusion	0.91	93.12
Concat. + MLP	0.84	**93.48**

(b) Ablation on modulated operators.

Modulated operators	#TP (M)	PB_T50_RS (%)
$\boldsymbol{B}_t$ only	0.74	92.08
$\boldsymbol{C}_t$ only	0.74	92.43
$\boldsymbol{\Delta}_t$ only	0.76	92.31
$\boldsymbol{B}_t, \boldsymbol{C}_t$	0.78	92.67
$\boldsymbol{B}_t, \boldsymbol{\Delta}_t$	0.80	<u>92.96</u>
$\boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\Delta}_t$	0.84	**93.48**

During training, each point cloud is instantiated with two valid serialization orders. The two branches share the backbone and SAA parameters and are jointly used to compute the task objective and the feature- and prediction-level consistency terms in DSCD. At inference, we retain only a single serialization branch and discard the auxiliary serialization used for consistency regularization. Therefore, Mantis does not require dual-pass inference, prediction ensembling, or test-time voting. All reported inference FLOPs and latency are measured under this single-serialization protocol and include the additional computation introduced by SAA. In contrast, the reported training time and peak memory account for the complete dual-serialization training procedure with DSCD.

B.2 Additional ablation study

Ablation study on different scanning curves. We further evaluate DSCD under different scanning curves, including Hilbert, Morton (Z-order), and their transposed variants. As shown in Table 5 (a), all structured scanning strategies consistently outperform random serialization on the hardest PB_T50_RS setting, indicating that geometry-aware traversal orders are important for stable state propagation in Mamba-based backbones. Moreover, since DSCD enforces consistency across two serialization branches, the scanning curves affect not only the sequence modeling quality within each branch, but also the complementarity between the two views. Among all candidates, using Hilbert and Trans-Hilbert as the two training views achieves the best performance, likely because it provides a favorable balance between locality preservation and cross-serialization diversity.

Ablation study on bottleneck dimension r. The SAA aims to reduce the number of trainable parameters during fine-tuning, and the bottleneck dimension r balances adaptation efficiency and downstream performance. As shown in Table 5 (b), while increasing r moderately enlarges the adaptation capacity of SAA, the gain over $r = 8$ remains marginal, suggesting that a small bottleneck is sufficient to capture most useful task-specific modulation. Therefore, we adopt $r = 8$ as the default setting, since it provides the best trade-off between accuracy and parameter efficiency.

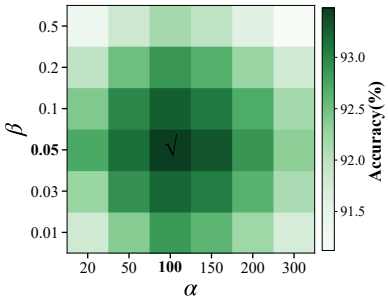


Figure 8: Sensitivity analysis of the DSCD hyperparameters α and β.

Ablation study on controller design. To examine whether the sparse proximal controller is necessary, we compare the proposed soft-thresholding controller with several dense or hard-selection alternatives under matched trainable parameters. As shown in Table 5 (c), replacing the soft-thresholding controller with Dense-MLP, Sigmoid Gate, Tanh Gate, or Hard-threshold consistently degrades performance. This indicates that the improvement of SAA lies not merely in low-rank dynamic parameterization, but also in sparse and magnitude-controlled perturbations induced by soft-thresholding. Therefore, the soft-thresholding controller provides a more stable way to modulate frozen Mamba backbones.

Table 7: Controlled comparison of state-aware adaptation and dual-serialization regularization on Mamba3D. LoRA (B, C, Δ) adapts native SSM projections under a matched parameter budget.

Methods	Dual-Ser.	DSCD	z_t	#TP (M)	PB_T50_RS
Full Fine-tuning Controls					
Full fine-tuning			–	16.90	92.05
Full fine-tuning + Dual-Ser.	✓		–	16.90	92.12
Full fine-tuning + DSCD	✓	✓	–	17.02	92.46
Conventional PEFT Controls					
Adapter			–	0.87	84.45
Adapter + DSCD	✓	✓	–	0.99	86.08
LoRA			–	0.91	83.19
LoRA (B, C, Δ)			–	0.91	89.41
LoRA (B, C, Δ) + DSCD	✓	✓	–	1.03	90.63
State-Aware Adaptation					
SAA w/o block context	✓			0.72	89.88
SAA	✓		✓	0.72	91.86
Mantis	✓	✓	✓	0.84	**93.48**

Table 8: Few-shot learning on ModelNet40 (Wu et al., 2015). We report the average accuracy (%) ± standard deviation (%) over 10 independent experiments without voting.

Methods	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
Self-Supervised Learning (Full Fine-tuning)				
Point-BERT (Yu et al., 2022)	94.6±3.1	96.3±2.7	91.0±5.4	92.7±5.1
MaskPoint (Liu et al., 2022)	95.0±3.7	97.2±1.7	91.4±4.0	93.4±3.5
Point-MAE (Pang et al., 2022)	96.3±2.5	97.8±1.8	92.6±4.1	95.0±3.0
ReCon (Qi et al., 2023)	97.3±1.9	98.9±1.2	93.3±3.9	93.3±3.9
PointGPT-S (Chen et al., 2023)	**98.0±1.9**	**99.0±1.0**	**94.1±3.3**	96.1±2.8
ACT (Dong et al., 2023)	96.8±2.3	98.0±1.4	93.3±4.0	95.6±2.8
Self-Supervised Learning (Parameter-Efficient Fine-tuning)				
PointMamba	96.9±2.0	99.0±1.1	93.0±4.4	95.6±3.2
+ Mantis	97.2±1.9	**99.0±1.0**	93.4±4.0	**96.4±3.1**
Mamba3D	96.4±2.2	98.2±1.2	92.4±4.1	95.2±2.9
+ Mantis	95.3±3.5	98.9±2.4	91.7±3.2	95.3±3.2
ZigzagPointMamba	96.0±2.1	99.0±1.2	90.0±2.2	94.2±1.0
+ Mantis	96.2±1.6	99.0±1.1	92.8±2.1	95.5±4.6

Ablation study on fusion function $\Phi(\cdot)$. We study choices for the fusion function $\Phi(\cdot)$ in SAA. As shown in Table 6 (a), simple addition or concatenation provides limited fusion capacity, while gated fusion and cross-attention achieve better results with more trainable parameters. In comparison, the proposed MLP-based fusion function achieves the best performance, suggesting that nonlinear fusion of the projected control features is beneficial for generating effective state-aware control signals.

Ablation study on modulated operators. We also examine which state-space operators should be modulated by SAA. As shown in Table 6 (b), modulating a single operator performs worse, indicating that adapting only one part of the selective SSM is insufficient. Jointly modulating $\boldsymbol{B}_t$, $\boldsymbol{C}_t$, and $\boldsymbol{\Delta}_t$ achieves the best performance, indicating that input injection, state readout, and adaptive discretization provide complementary adaptation capacity. We therefore retain the structured transition operator $\boldsymbol{A}$ frozen and apply SAA only to the native input-dependent selective coefficients.

Controlled analysis of state-aware adaptation. To disentangle the effects of state-aware control, dual-serialization training, and consistency regularization, we conduct controlled comparisons in Table 7. Full fine-tuning benefits marginally from dual serialization and moderately from DSCD, suggesting that the improvement of Mantis is not primarily caused by the dual-view training protocol. Adapting the native projections generating B, C, and Δ substantially improves LoRA, showing that conventional PEFT can indeed affect selective state dynamics when appropriately inserted. However, this SSM-oriented LoRA remains below SAA, even when equipped with DSCD. Furthermore, removing the incoming block context z_t from the pre-scan controller causes a noticeable degradation under the same parameter budget. Together, these results indicate that contextualized pre-scan conditioning strengthens SAA beyond input-only dynamic modulation, with DSCD providing regularization.

Hyperparameter sensitivity analysis. We analyze the sensitivity of DSCD to the loss weights α and β. Figure 8 shows that Mantis is relatively stable across a broad range of hyperparameters, indicating that the effectiveness of DSCD is not restricted to a single carefully tuned setting. The best performance occurs around $\alpha = 100$ and $\beta = 0.05$, which are adopted as the default hyperparameters.

B.3 Few-shot Learning

We further conduct few-shot experiments on ModelNet40(Wu et al., 2015) to assess our few-shot transfer learning ability. Following previous works(Pang et al., 2022), we employ the n-way, m-shot configuration with $n \in \{5, 10\}$ and $m \in \{10, 20\}$. As shown in Table 8, our Mantis achieves the best or second-best accuracy in all configurations, indicating its effectiveness in few-shot learning. This consistent trend across configurations highlights Mantis's ability to reduce performance variance, which is crucial for robust few-shot learning under limited data.

B.4 Detailed results for classification and part segmentation

To provide a more fine-grained evaluation, we report per-category results for both object classification and part segmentation. Table 9 reports category-wise IoU on ShapeNetPart(Yi et al., 2016), while

Table 9: Per-category part segmentation results on ShapeNetPart(Yi et al., 2016). We report instance-level mIoU, class-level mIoU, and category-wise IoU (%).

Methods	Inst.	Cls.	Air.	Bag	Cap	Car	Chair	Ear.	Guitar	Knife	Lamp	Laptop	Motor.	Mug	Pistol	Rocket	Skate.	Table
Full Fine-tuning																		
PointMamba	85.3	82.6	83.3	83.5	86.6	79.0	90.0	77.5	90.7	86.1	83.9	94.6	72.8	93.4	83.5	60.4	74.7	81.6
Mamba3D	85.6	83.6	83.7	84.3	87.5	76.5	91.2	80.6	91.2	87.5	87.1	95.6	72.2	94.2	83.7	66.5	72.5	83.4
ZigzagPointMamba	85.8	84.2	84.3	84.7	87.9	78.5	91.2	80.6	91.2	87.5	87.1	95.6	74.2	94.2	83.7	68.6	74.6	83.3
Parameter-Efficient Fine-tuning																		
PointMamba + Mantis	85.1	82.9	83.0	84.0	87.0	78.6	89.8	78.5	90.6	86.5	84.9	94.8	73.3	93.6	83.7	62.2	74.8	81.1
Mamba3D + Mantis	86.1	84.1	83.9	84.7	87.9	78.2	91.2	80.4	91.0	87.5	87.1	95.6	74.2	94.2	83.7	68.1	74.6	83.1
ZigzagPointMamba + Mantis	85.9	84.5	84.5	84.9	88.1	78.3	91.4	81.1	91.3	87.8	87.7	95.8	74.5	94.4	83.9	70.1	74.6	83.6

Table 10: Per-class classification accuracy on ScanObjectNN(Uy et al., 2019) (PB_T50_RS). We report mean class accuracy (mAcc), overall accuracy (OA), and class-wise accuracy (%).

Methods	mAcc	OA	Bag	Bin	Box	Cab.	Chair	Desk	Display	Door	Shelf	Table	Bed	Pillow	Sink	Sofa	Toilet
Full Fine-tuning																	
PointMamba	87.6	89.3	81.0	89.2	75.2	89.1	92.8	85.5	89.0	92.4	88.9	87.6	86.8	88.9	87.5	92.0	87.7
Mamba3D	90.2	92.1	83.8	92.0	78.2	91.6	95.5	88.2	91.4	95.2	91.6	90.2	89.4	91.3	89.8	94.6	90.2
ZigzagPointMamba	86.9	88.7	79.9	88.9	73.6	88.5	92.3	84.5	88.7	92.3	88.5	86.8	86.2	88.2	86.6	91.5	87.0
Parameter-Efficient Fine-tuning																	
PointMamba + Mantis	88.3	90.0	82.5	90.0	77.0	89.6	92.9	86.8	89.4	92.6	89.3	88.6	87.9	89.3	88.3	92.2	88.1
Mamba3D + Mantis	91.6	93.5	86.5	93.1	81.5	92.7	95.8	90.2	92.6	95.6	92.5	91.7	91.2	92.5	91.6	95.1	91.4
ZigzagPointMamba + Mantis	88.5	90.1	82.9	90.0	77.7	89.6	93.0	87.1	89.8	92.8	89.6	88.6	88.0	89.4	88.5	92.2	88.3

Table 10 presents class-wise classification accuracy on ScanObjectNN(Uy et al., 2019) (PB_T50_RS). The results show that the improvement of Mantis is not concentrated on a few categories, but is generally consistent across different object and part classes.

For part segmentation, Mantis maintains competitive instance-level mIoU and improves class-level mIoU across all three Mamba-based backbones. The gains on categories such as "Bag", "Cap", "Knife", "Lamp", "Mug", and "Rocket" suggest that Mantis is effective in preserving fine-grained local geometric structures under parameter-efficient adaptation. For object classification, Mantis consistently improves both mAcc and OA over the corresponding full fine-tuning baselines, with clear gains on challenging categories such as "Bag", "Box", "Desk", "Bed", "Pillow", and "Sink". These categories often exhibit larger intra-class variation or stronger geometric ambiguity, indicating that state-aware adaptation and cross-serialization consistency help improve robustness under noisy and order-sensitive real-world point cloud inputs.

B.5 Part segmentation visualization

In Figure 10, 11, and 12, we visualize our Mantis part segmentation results on the Mamba3D(Han et al., 2024) baseline. We present all 16 ShapeNetPart(Yi et al., 2016) categories from four different viewpoints. Mantis accurately segments large, well-defined parts such as airplane wings and chair seats, while also capturing small or complex parts like guitar strings and mug handles. The four-view visualizations demonstrate robustness to camera angles and occlusions, maintaining consistent part predictions. Even fine-grained structures are well-preserved, highlighting effective feature aggregation and state-aware adaptation. Color-coded part labels separate adjacent parts, reducing ambiguity. Despite achieving these visual results, Mantis introduces only a small number of trainable parameters, confirming its parameter-efficient design. Overall, these visualizations validate that Mantis provides accurate, robust, and efficient fine-grained part segmentation across diverse objects.

B.6 Additional Results on Large-scale Semantic Segmentation

Table 11 presents detailed results on ScanNet V2(Dai et al., 2017). Mantis achieves 50.4 VmIoU and 50.0 PmIoU with 0.73M trainable backbone parameters. Compared with existing PEFT methods, Mantis shows a consistent performance advantage while using a relatively small number of trainable parameters, and its performance remains close to full fine-tuning. The class-wise results further show that the gains are distributed across diverse semantic categories rather than concentrated on a few classes, reflecting the stable adaptation capability of Mantis across heterogeneous scene semantics.

Table 12 provides a detailed efficiency comparison on ScanNet v2. On PointMamba/Mamba3D, Mantis reduces trainable backbone parameters by 95.1%/95.3%, peak training memory by 6.8%/7.9%,

Table 11: Semantic segmentation results on the ScanNet V2 validation set(Dai et al., 2017). We report voxel-level mean IoU (VmIoU), point-level mean IoU (PmIoU), point-level per-class IoU (%), and trainable backbone parameters (M). Among all PEFT methods, the best and second-best results are highlighted in black and underlined, respectively.

Method	ScanNet V2 Val Dai et al. (2017)			wall	floor	cabinet	bed	chair	sofa	table	door	window	books.	picture	counter	desk	curtain	refrig.	shower	toilet	sink	bathtub	other
	Param.	VmIoU	PmIoU																				
Self-Supervised Learning (Full Fine-tuning)																							
Point-MAE (Pang et al., 2022)	21.82	51.2	50.8	70.8	91.1	47.5	63.8	66.0	55.4	52.3	38.1	35.8	47.8	28.0	33.4	34.6	46.8	51.7	41.3	73.1	48.5	59.1	30.9
ReCon (Qi et al., 2023)	21.82	50.8	50.5	70.4	90.4	43.3	64.5	65.0	53.8	52.2	37.2	36.0	46.9	27.9	33.5	32.2	44.7	54.2	43.2	75.3	49.4	57.5	32.4
Mamba3D (Han et al., 2024)	15.39 (100%)	51.3	50.7	69.8	87.6	43.2	62.7	67.3	54.4	54.7	40.2	35.7	45.9	29.4	34.5	38.0	50.0	54.8	41.1	71.0	45.6	57.8	30.3
General PEFT Methods																							
Mamba3D (lin.)	0.00 (0.00%)	40.3	39.8	60.6	79.5	34.8	50.0	55.6	45.6	41.5	29.0	26.5	35.5	16.9	22.8	23.1	36.7	41.4	32.2	61.0	36.6	47.2	19.5
+ Adapter (Houlsby et al., 2019)	1.90 (12.35%)	39.9 (-0.4)	39.5 (-0.3)	60.4	76.2	34.1	48.9	55.2	45.4	41.8	28.8	26.2	33.4	19.0	22.0	23.7	36.6	43.2	30.2	62.1	35.1	49.3	18.4
+ Prefix Tuning (Li & Liang, 2021)	1.67 (10.85%)	40.8 (+0.5)	40.4 (+0.6)	63.9	74.2	36.8	51.8	53.8	45.0	40.4	28.7	25.7	41.6	19.9	21.7	24.1	36.0	41.8	32.4	62.4	36.8	50.0	21.0
+ LoRA (Hu et al., 2022)	1.98 (12.87%)	42.7 (+2.4)	42.2 (+2.4)	63.6	82.4	39.8	57.3	56.1	49.1	45.9	32.3	30.6	41.3	24.2	27.4	24.1	43.1	42.1	43.0	47.7	44.0	27.1	22.9
PEFT Methods for Point Cloud																							
+ IDPT (Zha et al., 2023)	2.11 (13.71%)	39.6 (-0.7)	39.1 (-0.7)	57.0	75.5	33.7	53.0	56.3	45.5	40.9	26.2	27.7	37.4	15.8	18.7	21.7	39.8	42.3	28.8	60.2	35.1	47.7	18.7
+ DAPT (Zhou et al., 2024)	1.93 (12.54%)	41.5 (+1.2)	41.0 (+1.2)	62.0	80.7	31.4	51.0	54.9	47.8	43.4	30.7	26.9	37.0	25.2	20.5	23.3	38.2	41.1	33.5	61.7	40.8	48.9	21.0
+ PMA (Zha et al., 2025a)	1.77 (11.50%)	43.1 (+2.8)	42.3 (+2.5)	61.3	81.5	36.3	55.2	56.9	47.2	46.5	32.5	28.1	35.8	20.3	24.9	27.9	38.0	43.3	34.9	61.6	40.2	50.3	23.3
+ PointGST (Liang et al., 2025)	1.05 (6.82%)	43.2 (+2.9)	42.7 (+2.9)	62.6	83.0	34.9	55.2	55.3	48.6	42.3	31.4	34.7	35.2	27.8	23.4	26.8	36.7	46.1	33.7	64.0	40.6	54.4	17.3
+ PoinTriE (Sun et al., 2026c)	1.08 (7.02%)	43.9 (+3.6)	43.4 (+3.6)	62.5	89.0	35.9	55.8	56.5	49.8	44.7	32.0	31.7	36.5	21.8	23.9	27.4	38.4	45.5	37.7	63.1	39.9	54.1	21.8
+ Mantis	0.73 (4.74%)	50.4 (+10.1)	50.0 (+10.2)	70.3	88.4	44.5	62.3	64.4	55.1	52.6	38.3	36.2	45.4	27.1	32.2	34.5	47.3	52.4	41.5	71.2	47.4	58.7	30.2

Table 12: Efficiency comparison on ScanNet V2(Dai et al., 2017).

Method	Train. Backbone (M)	Head (M)	Inf. FLOPs (G)	Peak Mem. (GB)	AdamW States (GB)	Train Time (min/ep.)	Latency (ms/scene)	Storage (MB)	VmIoU (%)	PmIoU (%)
Point-MAE	21.82	5.20	64.8	19.2	0.216	19.4	91.0	108.1	51.2	50.8
+ PointLoRA (CVPR'25)	2.19	5.20	65.7	18.5	0.059	18.2	94.0	29.6	38.1	37.2
+ PMA (CVPR'25)	2.35	5.20	67.9	18.8	0.060	19.0	99.5	30.2	39.3	37.9
+ PoinTriE (ECCV'26)	1.90	5.20	68.7	18.8	0.056	19.3	100.6	27.6	43.4	42.1
PointMamba	12.15	6.13	41.5	14.8	0.146	14.5	64.5	73.1	50.3	50.1
+ PMA (CVPR'25)	1.17	6.13	46.8	14.4	0.058	16.7	73.2	29.2	42.7	41.6
+ Mantis (Ours)	**0.60**	**6.13**	**43.4**	**13.8**	**0.055**	**15.3**	**66.1**	**27.5**	**49.8**	**49.5**
Mamba3D	15.39	6.47	48.7	16.5	0.175	16.3	72.8	87.4	51.3	50.7
+ PMA (CVPR'25)	1.77	6.47	54.8	16.0	0.066	18.6	82.4	33.0	43.1	42.3
+ Mantis (Ours)	**0.73**	**6.47**	**51.1**	**15.2**	**0.058**	**16.9**	**75.0**	**28.8**	**50.4**	**50.0**

AdamW-state memory by 62.3%/66.9%, and per-task storage by 62.4%/67.0%, respectively. These substantial efficiency gains come with limited computational overhead, requiring only 1.9/2.4G additional FLOPs, 0.8/0.6 min per epoch, and 1.6/2.2 ms per scene. Meanwhile, Mantis maintains performance close to full fine-tuning and consistently improves over PMA, indicating a favorable overall balance between practical efficiency and downstream performance.

C Explanatory Experiments and Discussions

C.1 Training Dynamics and Optimization Stability

Observation: Token-Level Prompting Destabilizes Mamba Adaptation

Directly applying IDPT(Zha et al., 2023) to Mamba3D(Han et al., 2024) leads to unstable training and inferior convergence, suggesting that token-level prompt tuning is poorly aligned with Mamba's state-level sequence dynamics.

- **Mismatch:** IDPT is designed for Transformer-style token interactions, while Mamba relies on selective state-space dynamics and recurrent hidden-state propagation.
- **Mantis Design:** SAA adapts the state evolution directly, and DSCD regularizes predictions across different serialization orders.
- **Empirical Evidence:** Figure 9 illustrates that IDPT exhibits unstable training and slower convergence, whereas Mantis stabilizes training across challenging variants.

Mantis achieves smoother and more stable convergence, reaching performance comparable to or better than full fine-tuning with only about 5% of trainable parameters.

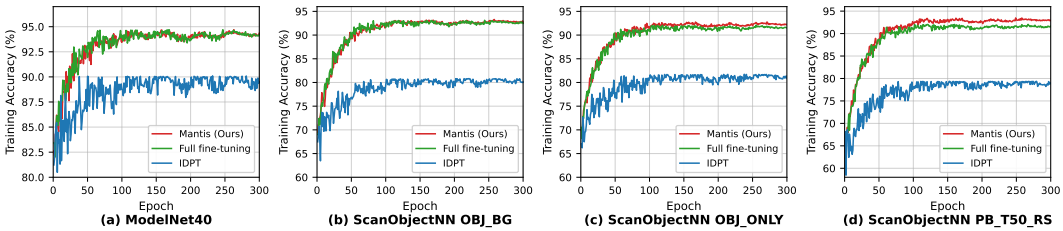


Figure 9: The classification accuracy curves of full fine-tuning, IDPT(Zha et al., 2023), and our Mantis with Mamba3D on ModelNet40(Wu et al., 2015) and ScanObjectNN(Uy et al., 2019).

Although IDPT(Zha et al., 2023) is effective for Transformer-based point cloud backbones, directly applying it to Mamba3D(Han et al., 2024) leads to unstable training dynamics and inferior convergence on downstream classification tasks. This observation is consistent with our motivation that token-level prompt tuning is not well aligned with the state-level sequence dynamics of Mamba. In contrast, Mantis introduces state-aware adaptation and dual-serialization consistency, which better match the selective state-space modeling mechanism. As a consequence, Mantis shows smoother and more stable convergence, especially on the challenging ScanObjectNN variants, and achieves final performance comparable to or better than full fine-tuning with only about 5% of trainable parameters.

C.2 Further analysis on PB_T50_RS

An interesting phenomenon in Table 1 is that the advantage of Mantis becomes more pronounced on the most challenging PB_T50_RS split than on the relatively easier OBJ_BG and OBJ_ONLY variants. We conjecture that this trend is closely related to the order sensitivity of Mamba-style sequence modeling on point clouds. Since point clouds are inherently unordered, the hidden-state propagation of a serialized selective SSM depends on the adopted traversal order. When the input is relatively clean and structurally regular, such order sensitivity may have only a limited effect. However, under more challenging real-world scans with stronger clutter, occlusion, and geometric corruption, different valid serializations are more likely to induce larger discrepancies in state propagation and feature aggregation. In this regime, the robustness benefit of Mantis becomes more evident.

To verify this explanation, we measure the feature-level and prediction-level cross-serialization discrepancies on ScanObjectNN variants using the same formulations as the consistency terms in Sec. 3.3. We compute both quantities on the test set and report their average values without the training weights α and β. Lower values indicate stronger consistency across serialization views.

Key Insight: Why Mantis Excels on PB_T50_RS

PB_T50_RS strongly exposes the order-sensitivity of frozen Mamba backbones in real-world point clouds. Mantis effectively addresses this through:

- **State-Aware Adapter (SAA):** Dynamically modulates state-space transitions to adapt hidden states under noisy or partially corrupted inputs.
- **Dual-Serialization Consistency Distillation (DSCD):** Aligns representations and predictions across different serialization orders, mitigating instability.
- **Empirical Evidence:** Table 13 shows that Mantis consistently reduces feature-level and prediction-level cross-serialization discrepancies, with the largest reduction on PB_T50_RS.

These effects stabilize downstream predictions and yield the largest gains on PB_T50_RS. This does not mean PB_T50_RS is easier. Rather, it simply highlights weaknesses of frozen Mamba backbones under noisy, order-sensitive inputs, where Mantis can be particularly effective.

As shown in Table 13, the baseline model exhibits the largest feature-level and prediction-level discrepancy on PB_T50_RS among the three ScanObjectNN variants, confirming that harder real-world point clouds indeed amplify serialization sensitivity. In contrast, Mantis consistently reduces both discrepancies across all three variants, with the most substantial reduction observed on PB_T50_RS. This observation supports our hypothesis that the stronger performance of Mantis on the hardest split mainly stems from improved robustness to serialization-induced instability.

Table 13: Cross-serialization discrepancy on different ScanObjectNN(Uy et al., 2019) variants.

Methods	OBJ_BG		OBJ_ONLY		PB_T50_RS	
	Feat. Disc.	Pred. Disc.	Feat. Disc.	Pred. Disc.	Feat. Disc.	Pred. Disc.
Mamba3D + Full Fine-tuning	0.148	0.031	0.136	0.027	0.221	0.054
Mamba3D + Mantis	0.109 (↓ 26.4%)	0.019 (↓ 38.7%)	0.101 (↓ 25.7%)	0.017 (↓ 37.0%)	0.128 (↓ 42.1%)	0.024 (↓ 55.6%)

From a modeling perspective, this behavior can be understood from the complementary effects of the two components in Mantis. On the one hand, SAA dynamically modulates the selective state-space dynamics conditioned on the current input, incoming block context, and serialization condition, enabling the frozen Mamba backbone to adapt its state evolution more flexibly under noisy and partially corrupted observations. On the other hand, DSCD explicitly regularizes the two serialization branches by aligning their feature representations and predictive distributions, thereby mitigating the instability caused by different traversal orders. When real-world perturbation and serialization ambiguity become more severe, the advantage of Mantis becomes particularly evident.

Importantly, this phenomenon should not be interpreted as PB_T50_RS being easier. Rather, it indicates that PB_T50_RS more strongly exposes the weakness of frozen Mamba backbones under noisy and order-sensitive inputs, thus creating a setting in which our robustness-oriented design is especially effective.

C.3 Discussion on the scope of Mamba-native adaptation

We further clarify the scope of "Mamba-native" adaptation in this work. PMA(Zha et al., 2025a) is an important 3D PEFT baseline that introduces Mamba into point cloud adaptation. However, its design motivation and adaptation target are different from ours. PMA is motivated by the observation that intermediate features of a frozen pre-trained model contain complementary information that is often discarded by conventional PEFT methods. Accordingly, PMA constructs an ordered sequence from intermediate-layer features and employs a Mamba adapter to fuse cross-layer semantics for downstream tasks. In its original formulation, the backbone is a frozen Transformer encoder, while Mamba is introduced as an orthogonal adapter for feature fusion. Notably, PMA focuses its evaluation on Transformer-style pre-trained backbones such as Point-BERT(Yu et al., 2022), Point-MAE(Pang et al., 2022), PointGPT(Chen et al., 2023), and ReCon(Qi et al., 2023), further reflecting its original scope as a feature-fusion adapter for existing PFMs rather than a method tailored to frozen Mamba backbone dynamics.

Clarification: Scope of Mamba-Native Adaptation

While PMA(Zha et al., 2025a) introduces Mamba modules for feature fusion on Transformer backbones, it does not adapt the frozen Mamba backbone itself. In contrast, Mantis directly modulates the native selective coefficients $(\boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\Delta}_t)$ inside frozen Mamba backbones, thereby adapting state evolution without altering the pretrained structured transition operator $\boldsymbol{A}$.

- **PMA:** Acts as an auxiliary adapter on intermediate features; evaluated on Transformer-style PFMs; cannot fully exploit Mamba backbone dynamics.
- **Mantis:** Targets the backbone itself, aligning PEFT with Mamba's state-level dynamics, achieving stronger performance on frozen Mamba backbones.
- **Implication:** Designing Mamba-native PEFT requires adaptation mechanisms matched to state-space modeling, rather than reusing Mamba modules for PEFT.

Overall, Mantis is not about using Mamba as an external module, but about serving Mamba backbones through state-level adaptation.

In contrast, Mantis targets frozen Mamba-based point cloud backbones themselves. Instead of inserting a Mamba module outside the backbone to aggregate intermediate features, Mantis adapts the selective state-space dynamics inside frozen Mamba backbones by modulating the native input-dependent coefficients $(\boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\Delta}_t)$ while retaining the structured transition operator $\boldsymbol{A}$ frozen. Therefore, we consider "Mamba-native 3D PEFT" not as the first use of Mamba modules in 3D PEFT, but as a state-level adaptation framework specifically tailored to frozen Mamba-based point cloud

backbones. This distinction can be summarized as follows: **PMA uses Mamba as an auxiliary adapter, whereas Mantis is designed for adapting Mamba backbones.**

This distinction also motivates our choice of PMA as a representative PEFT baseline. Since PMA is the closest existing 3D PEFT method involving Mamba modules, comparing with it allows us to examine whether a general Mamba-based adapter designed for feature fusion can be directly transferred to frozen Mamba backbones. The results show that such feature-level adapter fusion is not sufficient for Mamba-backbone adaptation. In contrast, Mantis consistently achieves stronger performance by directly modulating the internal state-space operators. This comparison highlights the necessity of designing PEFT mechanisms that are aligned with the state-level dynamics of Mamba, rather than only reusing Mamba as an external feature-fusion module.

D SYMBOL TABLE

Table 14: Symbol table

Symbol	Meaning
$P \in \mathbb{R}^{M\times 3}$	Input point cloud with M points.
$p \in \mathbb{R}^{n\times 3}$	Sampled key points after Farthest Point Sampling (FPS).
$p_h, p_{h'}$	Two valid serialized sequences of the same point cloud.
K	Number of nearest neighbors in each local patch.
d, d_h	Token feature dimension and hidden-state dimension.
d_o	Dimension of learnable order embedding.
d_ϕ	Dimension of the shared control space in SAA.
$\phi_t^{(k)}$	Fused control representation at step t for branch k.
$E_h^{(1)}, E_{h'}^{(2)}$	Token features under the two serialization orders.
$E^{(k)}$	Token features of branch k ($k \in \{1, 2\}$).
$o^{(k)}$	Learnable order embedding for branch k.
$e^{(k)}$	Order-aware representation for branch k.
$\mathcal{G}(\cdot)$	Lightweight transformation for order-aware representation.
$\tilde{Z}_l^{(k)}$	LN-projected feature in l-th encoder block.
$X_l^{(k)}$	Intermediate feature fed into selective SSM.
$Z_l^{(k)}$	Output of l-th encoder block.
$Z_0^{(k)}, Z_L^{(k)}$	Input/final feature of SAA-Mamba encoder.
$z^{(k)}$	Projected global embedding in DSCD (g_φ).
r	Bottleneck dimension in SAA.
α, β, τ	Feature/prediction loss weights and DSCD temperature.
$\mathbf{q}_t^{(k)}$	Driving vector before soft-thresholding.
$\boldsymbol{\lambda}_t^{(k)}$	Adaptive threshold vector.
$\mathbf{u}_t^{(k)}$	Sparse task-conditioned control signal.
$d_O^{\text{out}}, d_O^{\text{in}}$	Output and input dimensions of selective coefficient O.
ρ	Contraction bound of the frozen state transition operator.
$h_t^{(k,0)}$	Hidden state from frozen backbone.
$\delta h_t^{(k)}$	Hidden-state deviation under SAA.
$\varepsilon_{\text{tr}}, \varepsilon_B$	Bounds on discretized-transition and input-operator perturbations.
H, X	Upper bounds on hidden-state and input norms.
$\varphi(\cdot)$	Order modulation mapping.
$\Phi(\cdot)$	Control-feature fusion function in SAA.
$\mathcal{F}_{\text{SAA}}(\cdot)$	Selective SSM enhanced with SAA.
$g_\varphi(\cdot)$	Branch-shared projection head in DSCD.
$\boldsymbol{W}_{t,i}^{(k)}$	Selective transfer kernel from step i to t.
$\boldsymbol{W}_{t,i}^{(k,0)}$	Frozen backbone kernel without SAA.
$\Delta\boldsymbol{W}_{t,i}^{(k)}$	Perturbation of selective transfer kernel.
$\mathbf{U}_O, \mathbf{V}_O$	Coefficient-specific low-rank modulation matrices.
$\boldsymbol{\Gamma}_{O,t}^{(k)}$	Sparse low-rank factorized correction before native-layout mapping.
$\delta_{O,t}^{(k)}$	Native-layout correction applied to selective coefficient O.
$\mathcal{R}_O(\cdot)$	Mapping from the factorized correction to the native coefficient layout.
$\mathcal{N}_h^{(1)}, \mathcal{N}_{h'}^{(2)}$	Local neighborhoods under two serialization branches.
k	Branch index for dual serialization.
$x_t^{(k)}, h_t^{(k)}, y_t^{(k)}$	Input, hidden state, and output at step t.
$\ell^{(k)}, \pi^{(k)}$	Logits and softened prediction of branch k.
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}$	Discretized SSM operators (ZOH).
$\widehat{\boldsymbol{A}}_t^{(k)}, \widehat{\boldsymbol{B}}_t^{(k)}$	SAA-induced discretized transition and input operators.
$\boldsymbol{A}$	Frozen structured continuous-time transition operator.
$\boldsymbol{B}_t^{(k)}, \boldsymbol{C}_t^{(k)}, \boldsymbol{\Delta}_t^{(k)}$	Input-dependent selective coefficients at step t.
$z_t^{(k)}$	Incoming pre-scan block representation.
$\widetilde{\boldsymbol{B}}_t^{(k)}, \widetilde{\boldsymbol{C}}_t^{(k)}, \widetilde{\boldsymbol{\Delta}}_t^{(k)}$	Selective coefficients after SAA modulation.
$W_x^\phi, W_z^\phi, W_e^\phi, W_{\text{drv}}^\psi, W_{\text{gt}}^\psi$	Learnable projection matrices in the pre-scan SAA controller.
$\mathcal{L}_{\text{task}}, \mathcal{L}_{\text{feat}}, \mathcal{L}_{\text{pred}}$	Task loss, feature consistency, and prediction consistency loss.

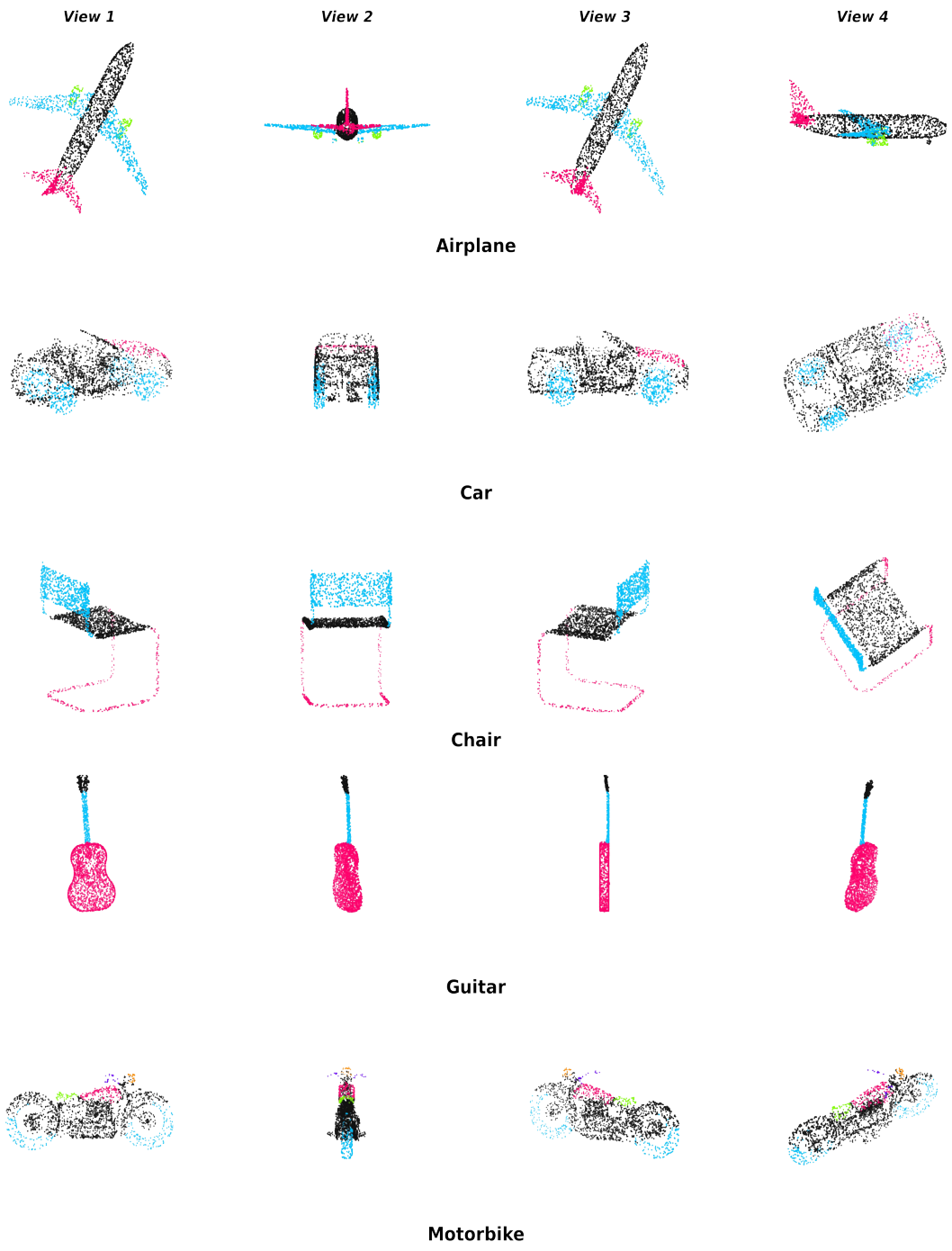


Figure 10: **Visualization results for part segmentation on ShapeNetPart(Yi et al., 2016)**. Projected prediction images from Mantis are shown across four different viewpoints, including the categories "Airplane", "Car", "Chair", "Guitar" and "Motorbike".

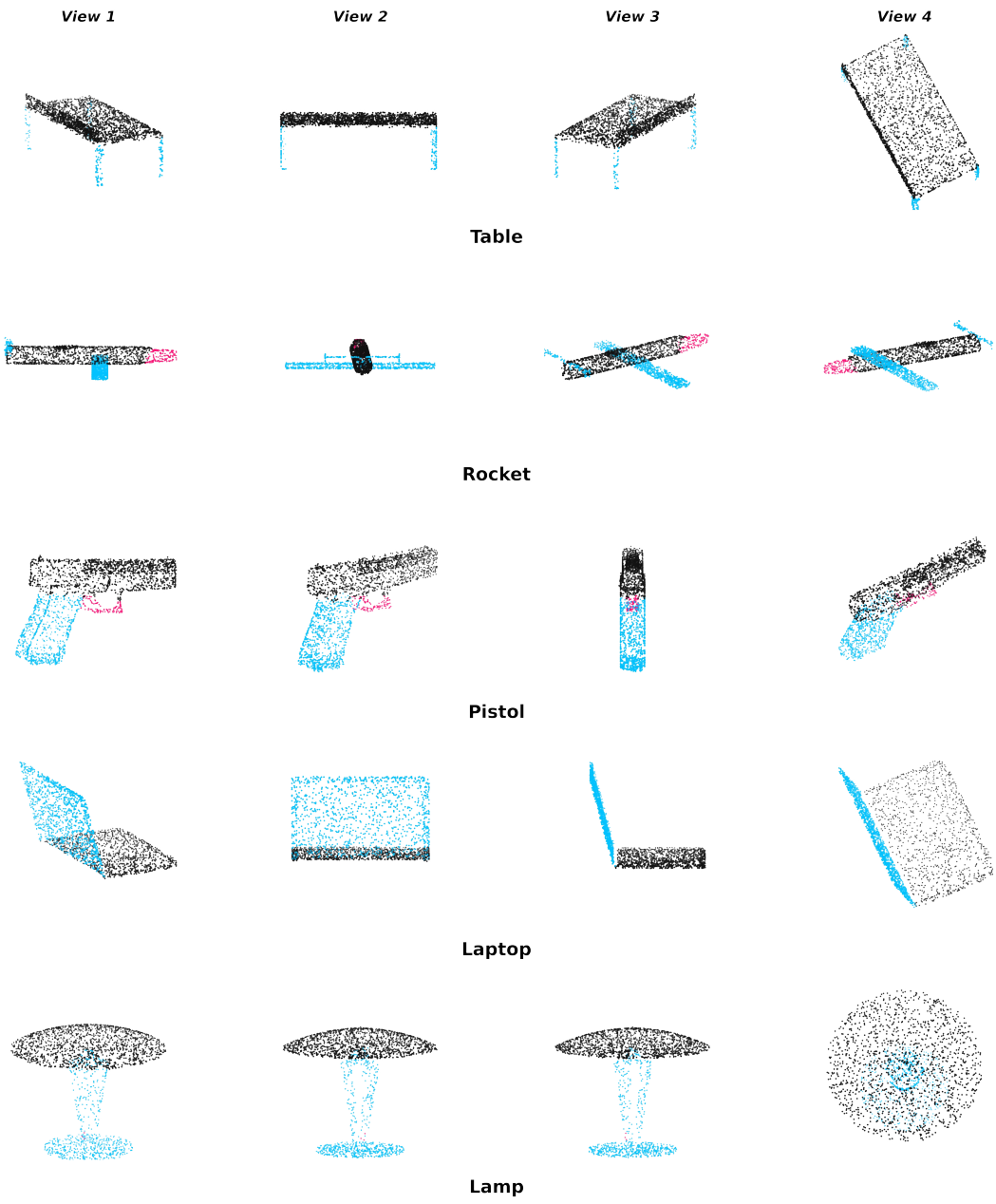


Figure 11: **Visualization results for part segmentation on ShapeNetPart(Yi et al., 2016)**. Projected prediction images from Mantis are shown across four different viewpoints, including the categories "Table", "Rocket", "Pistol", "Laptop" and "Lamp".

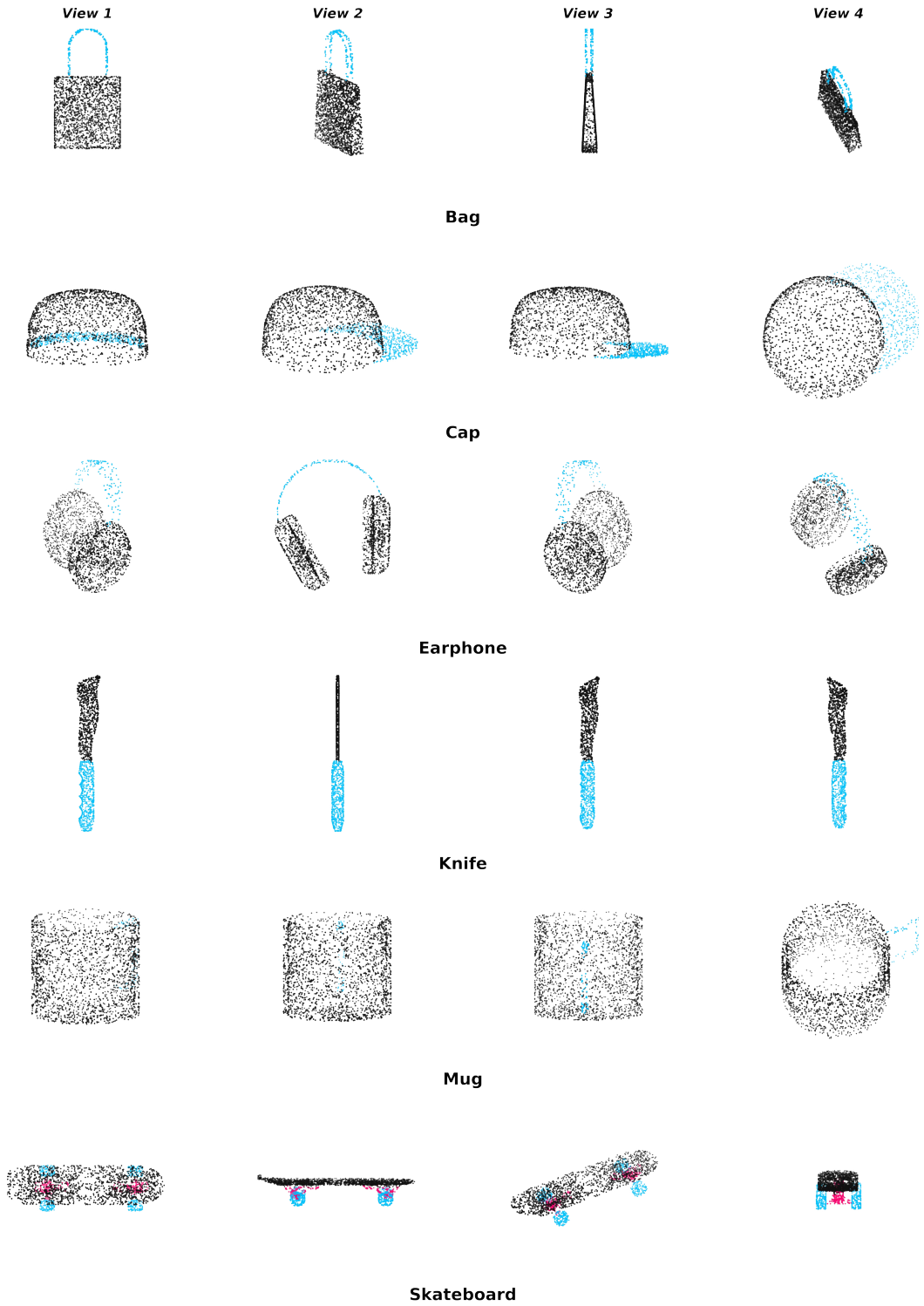


Figure 12: **Visualization results for part segmentation on ShapeNetPart(Yi et al., 2016)**. Projected prediction images from Mantis are shown across four different viewpoints, including the categories "Bag", "Cap", "Earphone", "Knife", "Mug" and "Skateboard".