

MAMBA3D: Enhancing Local Features for 3D Point Cloud Analysis via State Space Model

Xu Han*

Huazhong University of Science and Technology
Wuhan, China
xhanxu@hust.edu.cn

Zhaoxuan Wang

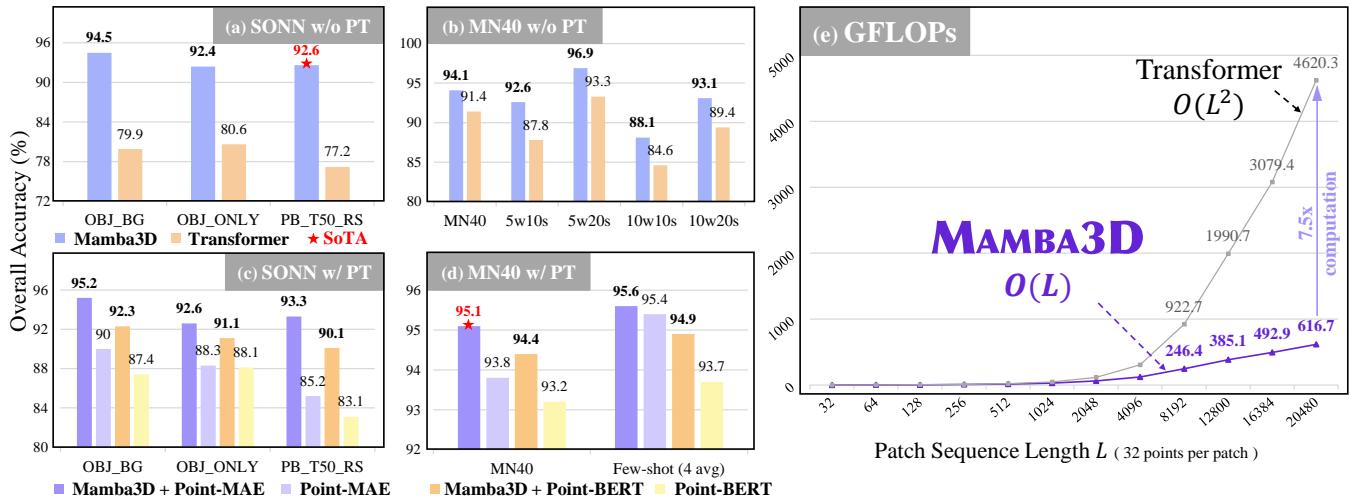
Huazhong University of Science and Technology
Wuhan, China
wang_zx@hust.edu.cn

Yuan Tang*

Huazhong University of Science and Technology
Wuhan, China
yuan_tang@hust.edu.cn

Xianzhi Li†

Huazhong University of Science and Technology
Wuhan, China
xzli@hust.edu.cn



1 Introduction

3D point cloud analysis serves as the foundation of wide-ranging applications such as autonomous driving [38, 48], VR/AR [19, 33], Robotics [46], etc. With the rich deep learning literature in 2D vision, a natural inclination is to develop deep learning methods for point cloud processing. Unlike 2D images, point clouds do not have a specific order and exhibit a complex geometric nature, which poses challenges for deep point cloud feature learning.

Starting from PointNet [36]/PointNet++ [37], deep learning on point clouds has gained popularity. A series of deep neural networks trained from scratch, such as DGCNN [53], PointMLP [31], PointNeXt [41] etc., are designed for robust point feature extraction. Recently, a flux of Transformer-based pre-training models [6, 10, 34, 39, 40, 62, 65] has been proposed to unleash the scalability and generalization of Transformer [51] for 3D point cloud representation learning, by leveraging a large amount of unlabelled data. However, the Transformer suffers from the dreaded quadratic bottleneck due to the pairwise communication brought by the attention mechanism. In other words, the Transformer-based model gets slower quadratically as the input size increases. Here, we focus on finding a new backbone for point cloud feature learning that achieves *superior performance, high efficiency, and scalability potential*.

The Mamba model [15], a recently proposed alternative to Transformer, is gaining attention for its efficiency. Built upon state space models (SSM), Mamba introduces a novel selection mechanism to effectively compress context, enabling it to handle long sequences. Also, the hardware-accelerated scan enables Mamba to achieve near-linear complexity during training. However, the straightforward adoption of Mamba does not achieve satisfactory performance on point cloud tasks due to the following challenges. Firstly, its recurrent/scan mode leads to sequential dependency that is unsuitable for unordered point clouds, causing unstable pseudo-order reliance. Secondly, Mamba lacks explicit local geometry extraction, which is crucial in point cloud learning [31, 35].

Driven by the above analysis, we present **MAMBA3D**, a novel state space model tailored for point cloud learning. Going beyond existing works, two essential technical contributions are delivered: (1) *Local Norm Pooling (LNP)*: a local feature extraction block comprising K-norm and K-pooling operators for local feature propagation and aggregation, respectively. To ensure the efficiency and scalability of our MAMBA3D, we design our LNP block as simple yet effective, utilizing only 0.3M parameters. (2) *Bidirectional-SSM (bi-SSM)*: a token forward SSM and a novel backward SSM that operates on the feature channel to obtain better global features. Considering the disorder of the point token sequence, we propose to treat the feature channel as an ordered sequence, which is more reliable and stable. Based on the original token forward (L+) SSM, we further design a feature reverse backward (C-) SSM to alleviate pseudo-order reliance, thus fully exploiting the global features.

Note that, there are concurrent works PointMamba [26] and PCM [66] that also apply Mamba to 3D point clouds. However, PointMamba ignores local feature extraction, while PCM does not support pre-training and is computationally intensive (2× parameters, 12× FLOPs). In contrast, MAMBA3D yields more representative point features by explicitly incorporating local geometry. Particularly, its linear complexity and large capacity allow for both training

from scratch, and equipping with various pre-training strategies, facilitating downstream tasks with promising performance.

To thoroughly evaluate MAMBA3D’s capacity and representation learning ability, we conduct extensive experiments by training our model from scratch, as well as pre-training using two different pre-training strategies following Point-BERT [62] and Point-MAE [34], respectively. Results show that MAMBA3D substantially surpasses both the Transformer-based counterparts and the two concurrent works on various downstream tasks, while having fewer parameters and FLOPs. For example, as shown in Fig. 1(a) and Fig. 1(b), MAMBA3D achieves **92.6%** overall accuracy (OA) on the ScanObjectNN [50] classification task, setting new SoTA among models trained from scratch, and outperforms Transformer on both ModelNet40 [57] object classification task and few-shot classification task. Similarly, as shown in Fig. 1(c) and Fig. 1(d), when equipped with the pre-training strategies proposed in Point-BERT and Point-MAE, MAMBA3D still outperforms Transformer on various tasks. Particularly, MAMBA3D achieves **95.1%** OA on ModelNet40, setting new SoTA among single-modal pre-trained models. Meanwhile, MAMBA3D reduces **30.8%** in parameters and **23.1%** in FLOPs compared to Transformer. Section 4 presents more experimental results.

In summary, this work makes the following contributions:

- We introduce MAMBA3D, a state space model with local geometric features tailored for point cloud learning, achieving superior performance with linear complexity.
- We design a Local Norm Pooling (LNP) block, enhancing local geometry extraction with only 0.3M parameters.
- We propose C-SSM, a feature reverse SSM, alleviating pseudo-order reliance in unordered points.
- Extensive experiments demonstrate MAMBA3D’s superior performance over Transformer, achieving multiple SoTA results and robust few-shot learning capabilities.

2 Related Work

2.1 Deep Point Cloud Learning

As deep neural networks (DNNs) continue to advance, point cloud feature learning has gained increasing attention, leading to the development of numerous deep architectures and models in recent years. Inspired by early models PointNet [36] and PointNet++ [37], some attempts [2, 8, 24, 25, 44, 53, 67] design various deep architectures to better capture local context information. Later, Transformer-inspired [51] models such as Point Transformers [55, 56, 68] and Stratified Transformer [23] have become popular backbones, fusing local and global features to achieve state-of-the-art results. However, these dedicated architectures for 3D understanding excel in specific tasks but struggle with transferring across tasks and modalities.

To fully make use of the massive unlabelled data, self-supervised pre-training thereby becomes a viable technique. For example, Point-BERT [62], Point-MAE [34], and MaskPoint [27] propose to pre-train the Transformer [51] with masked point modeling approaches [28, 49, 63, 65, 69]. These methods enable models to learn generalizable features, which are transferable to different tasks effectively. There are also multi-modal pre-training strategies like ACT [10], ULIP-2 [59], and ReCon [40] that leverage cross-modal information from language and images to enhance generalization

and robustness. However, their Transformer-based backbones suffer from quadratic complexity, posing challenges in handling long sequences, resulting in coarse-grained patching and information loss. In contrast, MAMBA3D leverages Mamba's linear complexity, surpassing Transformer in both performance and efficiency.

2.2 State Space Models

State Space Models (SSM) [16, 17, 22], inspired by continuous systems, have emerged as promising models for sequence modeling. Notably, S4 [16] demonstrates the ability to capture long-range dependencies with linear complexity, showcasing effectiveness across diverse domains like audio [13] and vision [32]. The newly proposed Mamba model [15] further improves upon S4 by introducing a selection mechanism. By parameterizing SSM based on inputs, Mamba selectively retains relevant information, facilitating efficient processing of long-sequence data.

To adapt Mamba from sequence data to unordered point cloud data, there are concurrent works PointMamba [26] and PCM [66]. PointMamba applies Mamba directly without considering the local contexts. Based on PointMLP [31], PCM lacks pre-training and suffers from excessive parameters, limiting Mamba's efficiency. In contrast, MAMBA3D effectively applies Mamba to point cloud learning, integrating the unique local geometry of point clouds. Additionally, we employ various pre-training strategies to validate MAMBA3D's scalability and large capacity.

3 Method

We aim to leverage Mamba for point cloud feature learning, emphasizing both global receptive field and local geometric details. Below, we first briefly review the State Space Models (SSM) and the Mamba model (Section 3.1), then followed by an overview and detailed explanations of our key designs (Section 3.2–Section 3.4), also with two pre-training strategies used here (Section 3.5).

3.1 Preliminaries: SSM and Mamba

Drawing from continuous systems, state space models (SSM) map input x_t to output y_t via a latent state h_t , with the state evolving over time t continuously. In practice, to accommodate discrete data like text and images, the SSM must be discretized first:

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \quad y_t = \bar{C}h_t, \quad (1)$$

where \bar{A} , \bar{B} , and \bar{C} are the discrete state, control, and output matrix, respectively. Because sequential parameters \bar{A} , \bar{B} , and \bar{C} exhibit Linear Time Invariance (LTI), we can parallelize the recurrent SSM in Eq. (1) using a convolutional method into Eq. (2):

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

where L is the length of the input sequence x , and $\bar{K} \in \mathbb{R}^L$ denotes a global convolution kernel, which can be efficiently pre-computed.

S4 [16] enhances SSM's ability for long sequence modeling and speed. Mamba [15] acknowledges the efficiency of LTI models like S4 in compressing extensive contexts into compact states compared to Transformer [51], which exhibits quadratic complexity during training due to zero-compression. However, the constant dynamics of LTI models, such as the input-independent parameters \bar{A} , \bar{B} , and \bar{C} in Eq. (1), limit their ability to selectively remember or

forget relevant information, constraining their contextual awareness. To enhance content-aware reasoning, Mamba introduces a selection mechanism to control how information propagates or interacts along the sequence dimension. This is achieved by making the parameters that affect interactions along the sequence input-dependent, as defined in Eq. (3):

$$h_t = s_{\bar{A}}(x_t)h_{t-1} + s_{\bar{B}}(x_t)x_t, \quad y_t = s_{\bar{C}}(x_t)h_t, \quad (3)$$

where $s_{\bar{A}}(x_t)$, $s_{\bar{B}}(x_t)$ and $s_{\bar{C}}(x_t)$ typically denote three linear projections applied to input x_t . The selection mechanism addresses the limitations of LTI models but makes parallelization shown in Eq. (2) impractical. To tackle this challenge, Mamba introduces a hardware-aware selective scan to achieve near-linear complexity. Please refer to the original Mamba paper [15] for further details.

3.2 MAMBA3D Overview

Though Mamba produces astounding results in sequential data, it is not straightforward to adapt Mamba to the 3D point cloud. On the one hand, Mamba employs recurrent/scan structure, which implies constant-time inference and linear-time training due to the effective context compression, but exhibits a unidirectional reliance. While this recurrent model works well for text, it poses challenges when dealing with unordered point clouds. On the other hand, Mamba's global receptive field cannot adequately capture local point geometry, limiting its ability to learn fine-grained features. To address the above issues, we introduce MAMBA3D, featuring an effective local norm pooling (LNP) block for explicit local geometry extraction and a specialized bidirectional SSM (bi-SSM) tailored for unordered points. Fig. 2(a) shows an overview of MAMBA3D.

3.2.1 Patch Embeddings. Given an input point cloud $P \in \mathbb{R}^{N \times 3}$ with N points, similar to existing works [34, 62], we first employ Farthest Point Sampling (FPS) to select L central points $P_C \in \mathbb{R}^{L \times 3}$. Then, for each central point P_C^i , we construct a local patch $x_P^i \in \mathbb{R}^{K \times 3}$ using K -Nearest Neighborhood (KNN) in P . Finally, we employ a light-PointNet [36] to extract features for each local patch, serving as its initial patch embeddings.

3.2.2 MAMBA3D Encoder. After obtaining the patch embeddings, they are treated as token sequences in the Transformer [51]. Similar to ViT [11] and BERT [9], we first introduce a learnable [CLS] token to aggregate information across the entire sequence. Then, we add standard learnable positional encoding [51] to these $L + 1$ tokens. The sequence is then fed into the MAMBA3D encoder for high-level feature embedding, which is finally connected to a simple fully connected layer for various downstream tasks.

The design of our encoder is illustrated in the right-most part of Fig. 2(a). Specifically, inspired by MetaFormer [61], we employ a Transformer-like structure consisting of a token mixer (i.e., LNP) and a channel mixer (i.e., bi-SSM) to extract local and global features, respectively. Each block is preceded by a Layernorm [3] and followed by a residual connection [21].

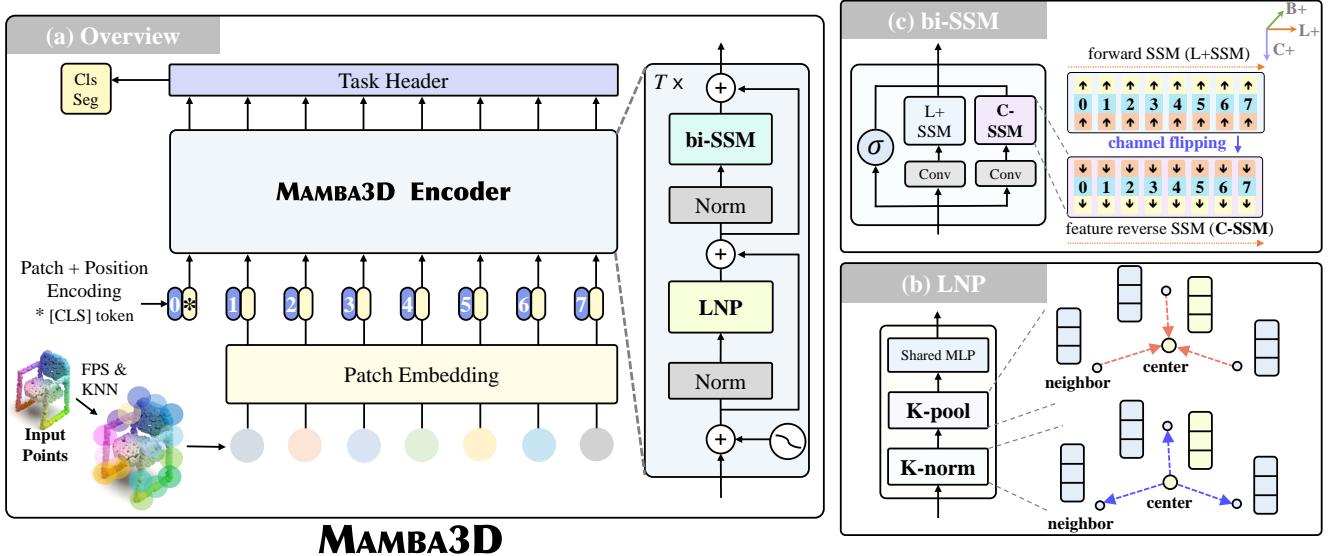


Figure 2: Illustration of MAMBA3D. (a) Overview. We first segment input point cloud into L patches using FPS & KNN, and then obtain the initial patch embeddings via a light-PointNet. After adding a [CLS] token, we apply standard positional encoding to all patch embeddings, which are then fed into MAMBA3D Encoder. Finally we use a task header to fit downstream tasks. (b-c) MAMBA3D Encoder details. The illustration of the MAMBA3D Encoder is inspired by Dosovitskiy et al. [11].

Overall, the pipeline of point embedding and encoder layer is represented by the following equations:

$$\mathbf{z}_0 = [\mathbf{x}_{cls}; \mathbf{x}_p^1 E; \mathbf{x}_p^2 E; \dots; \mathbf{x}_p^L E] + \mathbf{E}_{pos}, \quad (4)$$

$$\mathbf{z}'_\ell = \text{LNP}(LN(\mathbf{z}_{\ell-1} + \mathbf{E}_{pos})) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \dots T \quad (5)$$

$$\mathbf{z}_\ell = \text{bi-SSM}(LN(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \quad \ell = 1 \dots T \quad (6)$$

where \mathbf{z} is the output of each layer, $\mathbf{x}_{cls} \in \mathbb{R}^{1 \times C}$ is the learnable [CLS] token, E is the light-PointNet to project input patches from $\mathbf{x}_p \in \mathbb{R}^{L \times K \times 3} \mapsto \mathbf{x}_p E \in \mathbb{R}^{L \times C}$, and LN denotes the Layernorm operation. In practice, we stack T encoder layers, and a standard learnable positional encoding $\mathbf{E}_{pos} \in \mathbb{R}^{(L+1) \times C}$ is incorporated into every encoder layer, as in Point-MAE [34], to enhance the model's spatial awareness.

3.3 Local Norm Pooling

Local geometric features have been proven to be vital for point cloud feature learning, but were unfortunately ignored in Point-Mamba [26]. Typically, local features in point clouds are obtained by constructing a local graph using KNN, followed by feature fusion [31, 36, 37]. To ensure both effectiveness and efficiency, we design a novel Local Norm Pooling (LNP) block by simplifying the local feature extraction into two key steps: feature propagation and aggregation. Specifically, as illustrated in Fig. 2(b), LNP comprises two operators K-norm (propagation) and K-pooling (aggregation), alongside a shared MLP for channel alignment.

3.3.1 K-norm: propagation. After constructing a local graph with k neighbors using KNN around each central point, the feature propagation involves (1) enabling neighboring points to perceive relative features concerning the central point and (2) conducting feature fusion to update their features accordingly. To achieve this,

we first normalize the neighbor features $\mathbf{F}_K \in \mathbb{R}^{L \times k \times C}$ to get $\tilde{\mathbf{F}}_K \in \mathbb{R}^{L \times k \times C}$ as defined in Eq. (7). Then, we concatenate $\tilde{\mathbf{F}}_K$ with the repeated (by K times) central point feature $\mathbf{F}_C \in \mathbb{R}^{L \times k \times C}$ and apply a learnable linear transformation across the local graph to obtain the propagated features $\widehat{\mathbf{F}}_K \in \mathbb{R}^{L \times k \times 2C}$:

$$\tilde{\mathbf{F}}_K = \frac{\mathbf{F}_K - \mathbf{F}_C}{\sqrt{Var(\mathbf{F}_K - \mathbf{F}_C) + \epsilon}}, \quad \widehat{\mathbf{F}}_K = [\tilde{\mathbf{F}}_K \oplus \mathbf{F}_C] * \gamma + \beta, \quad (7)$$

where γ and β are trainable scale and shift vectors as in Layer-norm [3], respectively, and \oplus signifies feature channel concatenation. This linear transformation preserves topological features while capturing the rigid transformation of the local graph features. As depicted in the lower half of Fig. 2(b), our K-norm facilitates local feature propagation from the central point to its neighbors.

3.3.2 K-pooling: aggregation. After propagating features within the local graph, we aggregate the information back to the central point for feature updating. While max-pooling is commonly used for feature aggregation in unordered points to maintain order invariance [36], it would lead to information loss. Inspired by Softmax, we introduce K-pooling to efficiently perform local feature aggregation while mitigating this information loss, as defined in Eq. (8):

$$\widehat{\mathbf{F}}_C = \sum_i^K \frac{\exp \widehat{\mathbf{F}}_K^i}{\sum_j^K \exp \widehat{\mathbf{F}}_K^j} \cdot \widehat{\mathbf{F}}_K^i, \quad (8)$$

where $\widehat{\mathbf{F}}_C$ is the updated central point features. K-pooling maps from $\widehat{\mathbf{F}}_K \in \mathbb{R}^{L \times k \times 2C} \mapsto \widehat{\mathbf{F}}_C \in \mathbb{R}^{L \times 2C}$, generates updated central point features, as depicted in the upper half of Fig. 2(b).

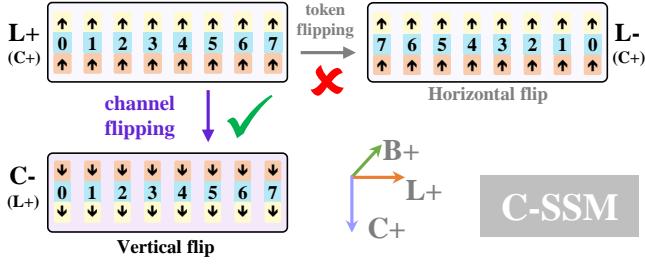


Figure 3: Illustration of C-SSM. Instead of horizontal token flip, we propose a vertical feature flip to alleviate the pseudo-order reliance.

Intuitively, the LNP block constructs a local graph with adjacent patches and facilitates local feature propagation and aggregation, enabling information exchange within the local field, thus capturing the geometric and semantic features of the local patches. The receptive field of the LNP is smaller than that of SSM. By adjusting the size of the receptive field, the LNP integrates local and global information, enabling MAMBA3D to more comprehensively capture the semantic information of 3D objects.

3.4 Bidirectional-SSM

Mamba is originally designed as a unidirectional model suited for processing 1D sequences like text. However, vision tasks often require an understanding of global spatial information. Hence, Vision Mamba [70] proposed using both forward and backward SSM to incorporate global information by simply horizontally flipping the token order, as illustrated by the L^+ and L^- embeddings in Fig. 3. However, unlike the structured grid of images, point clouds are unordered and irregular, learning sequence order causes an unreliable and unstable pseudo-order dependency. To address this, we instead propose to prioritize modeling the intrinsic distribution of feature vectors rather than point tokens. Specifically, we propose a novel backward SSM, named feature reverse SSM, or C-SSM, as illustrated in Fig. 3. Combining this with the original forward SSM in Mamba, termed L^+ -SSM, results in our bidirectional-SSM block, or bi-SSM for short. Formally, the bi-SSM block is defined as:

$$\text{bi-SSM}(\mathbf{F}) = \mathbf{F} + [\mathbf{L}^+\text{-SSM}(\mathbf{F}^{L^+})] + [\mathbf{C}^-\text{-SSM}(\mathbf{F}^{C^-})]. \quad (9)$$

With the input forward embedding \mathbf{F} , also denoted as \mathbf{F}^{L^+} , we perform a channel flip to obtain \mathbf{F}^{C^-} , which is then fed into L^+ -SSM and C-SSM block to generate the output embedding, respectively.

As shown in Fig. 3, we employ a vertical flip to obtain \mathbf{F}^{C^-} , instead of a horizontal flip to get \mathbf{F}^{L^-} . This approach reduces the pseudo-order reliance, crucial in unordered point clouds where token order lacks consistent meaning. For instance, two adjacent tokens might represent the tail and head of an airplane, respectively, which are spatially and semantically distant, posing challenges for continuous and complete feature learning. Mamba’s feature selection mechanism may exacerbate this, scattering features in high-dimensional space. Instead, by reversing the feature channel, the model prioritizes learning the distribution of feature vectors. These two directions—forward token embeddings and backward feature channels—carry distinct and more reliable information, enhancing MAMBA3D’s ability to acquire more effective knowledge.

3.5 Pre-training Details

Our MAMBA3D can not only be trained from scratch, but also be pre-trained with various pre-training strategies, thus facilitating downstream tasks with promising performance. In experiments, we verify MAMBA3D with two commonly used pre-training strategies proposed by Point-BERT [62] and Point-MAE [34].

3.5.1 Point-BERT pre-training strategy. Firstly, we randomly mask out 55%~85% input point embeddings, instead of a mask ratio between [0.25, 0.45] in Point-BERT. Increasing the mask ratio can not only speed up the training process, but also push MAMBA3D’s feature learning ability, enabling the model to learn from limited inputs. Then the MAMBA3D encoder processes both visible and masked embeddings to produce a token sequence. Meanwhile, we employ the pre-trained dVAE [45] weight of Point-BERT directly to predict token sequence from point embeddings as token guidance. Lastly, we calculate the L1 loss between the encoder’s output token sequence and the one from dVAE as the loss function.

3.5.2 Point-MAE pre-training strategy. Following Point-MAE, we use a masked point modeling approach and directly reconstruct masked points. We employ an encoder-decoder architecture, where the encoder processes only visible tokens and generates their encoding. Unlike Point-MAE, our decoder employ a different architecture from encoder, containing only bi-SSM block but no LNP block, which can speed up convergence without performance loss. The encoded visible tokens and masked tokens are fed into the decoder to predict masked points. Loss is calculated using the Chamfer Distance [12] between output and ground truth points. In downstream tasks, we only use the pre-trained encoder to extract features, with task headers appended for fine-tuning.

4 Evaluation

In this section, we first introduce the network implementation details. Then we evaluate MAMBA3D against various existing methods in multiple downstream tasks, including object classification, part segmentation, and few-shot learning. Finally, we show the results of the ablation study for our model.

4.1 Implementation Details

We employ $T=12$ encoder layers with feature dimension $C=384$, and set $k=4$ in the LNP block. During pre-training, we utilize the ShapeNet dataset [4], which contains ~50K 3D CAD models covering 55 object categories. Each input point cloud, containing $N=1024$ points, is divided into 64 patches with each consisting of 32 points. Pre-training employs the AdamW optimizer [30] with cosine decay, an initial learning rate of 0.001, a weight decay of 0.05, a dropout rate of 0.1, and a batch size of 128 for 300 epochs. During fine-tuning, the point cloud is divided into 128 patches, and we train the model with the AdamW optimizer with cosine decay, an initial learning rate of 0.0005, a weight decay of 0.05, and a batch size of 32 for 300 epochs. Unless specified, we use the same task header as Point-MAE [34] in all downstream tasks. When training from scratch, we use the same settings as in fine-tuning. All experiments are conducted using an NVIDIA RTX 3090 GPU.

Table 1: Classification results on the ScanObjectNN and ModelNet40 datasets. The inference model parameters #P (M), FLOPs #F (G), and overall accuracy (%) are reported. We compare with methods using the \triangle hierarchical Transformer architectures (e.g., Point-M2AE [65]), \bullet plain Transformer architectures, \circ dedicated architectures for 3D understanding, and \star Mamba-based architectures. \dagger means additional tuning. PT: pre-training strategy. Results of other models are before 0:00 UTC on April 1, 2024.

Method	PT	#P \downarrow	#F \downarrow	ScanObjectNN			ModelNet40 1k P \uparrow
				OBJ_BG \uparrow	OBJ_ONLY \uparrow	PB_T50_RS \uparrow	
<i>Supervised Learning Only: Dedicated Architectures</i>							
\circ PointNet [36]	\times	3.5	0.5	73.3	79.2	68.0	89.2
\circ PointNet++ [37]	\times	1.5	1.7	82.3	84.3	77.9	90.7
\circ DGCNN [53]	\times	1.8	2.4	82.8	86.2	78.1	92.9
\circ PointCNN [25]	\times	0.6	-	86.1	85.5	78.5	92.2
\circ DRNet [42]	\times	-	-	-	-	80.3	93.1
\circ SimpleView [14]	\times	-	-	-	-	80.5 \pm 0.3	93.9
\circ GBNet [43]	\times	8.8	-	-	-	81.0	93.8
\circ PRA-Net [7]	\times		2.3	--	-	81.0	93.7
\circ MVTN [20]	\times	11.2	43.7	92.6	92.3	82.8	93.8
\circ PointMLP [31]	\times	12.6	31.4	-	-	85.4 \pm 0.3	94.5
\circ PointNeXt [41]	\times	1.4	3.6	-	-	87.7 \pm 0.4	94.0
\circ P2P-HorNet [54]	\checkmark	-	34.6	-	-	89.3	94.0
\circ DeLA [5]	\times	5.3	1.5	-	-	90.4	94.0
<i>Supervised Learning Only: Transformer or Mamba-based Models</i>							
\bullet Transformer [51]	\times	22.1	4.8	79.86	80.55	77.24	91.4
\triangle PCT [18]	\times	2.9	2.3	-	-	-	93.2
\star PointMamba [26]	\times	12.3	3.6	88.30	87.78	82.48	-
\star PCM [66]	\times	34.2	45.0	-	-	88.10 \pm 0.3	93.4 \pm 0.2
\triangle SPoTr [35]	\times	1.7	10.8	-	-	88.60	-
\triangle PointConT [29]	\times	-	-	-	-	90.30	93.5
\star MAMBA3D w/o vot.	\times	16.9	3.9	92.94	92.08	91.81	93.4
\star MAMBA3D w/ vot.	\times	16.9	3.9	94.49	92.43	92.64	94.1
<i>with Self-supervised Pre-training</i>							
\bullet Transformer [51]	OcCo [52]	22.1	4.8	84.85	85.54	78.79	92.1
\bullet Point-BERT [62]	IDPT [64]	22.1+1.7 \dagger	4.8	88.12	88.30	83.69	93.4
\bullet MaskPoint [27]	MaskPoint	22.1	4.8	89.30	88.10	84.30	93.8
\star PointMamba [26]	Point-MAE	12.3	3.6	90.71	88.47	84.87	-
\bullet Point-MAE [34]	IDPT [64]	22.1+1.7 \dagger	4.8	91.22	90.02	84.94	94.4
\triangle Point-M2AE [65]	Point-M2AE	15.3	3.6	91.22	88.81	86.43	94.0
\bullet Point-BERT [62]	Point-BERT	22.1	4.8	87.43	88.12	83.07	93.2
\star MAMBA3D w/o vot.	Point-BERT	16.9	3.9	92.25 +4.82	91.05 +2.93	90.11 +7.04	94.4 +1.2
\bullet Point-MAE [34]	Point-MAE	22.1	4.8	90.02	88.29	85.18	93.8
\star MAMBA3D w/o vot.	Point-MAE	16.9	3.9	93.12 +3.10	92.08 +3.79	92.05 +6.87	94.7 +0.9
\star MAMBA3D w/ vot.	Point-MAE	16.9	3.9	95.18 +5.16	92.60 +4.31	93.34 +8.16	95.1 +1.3

4.2 Comparison on Downstream Tasks

We show MAMBA3D's results on downstream tasks here. For each experiment, we report results for models trained from scratch, as well as those employing two pre-training strategies. Unless specified, the results for MAMBA3D do not use a voting strategy.

4.2.1 Object Classification. We conduct classification experiments on both the real-world ScanObjectNN [50] dataset and the synthetic ModelNet40 [57] dataset.

Settings. ScanObjectNN dataset contains \sim 15K objects from 15 classes, scanned from the real world with cluttered backgrounds. We experiment with its three variants: OBJ_BG, OBJ_ONLY, and PB_T50_RS. We use *rotation* as data augmentation [10], with a point cloud size $N=2048$. ModelNet40 dataset includes \sim 12K synthetic 3D CAD models across 40 classes. We use $N=1024$ points as input, and apply *scale&translate* for data augmentation [36].

Table 2: Few-shot classification on ModelNet40 dataset. Overall accuracy (%) without voting is reported. *P-B* and *P-M* represent Point-BERT and Point-MAE strategy, respectively.

Method	5-way		10-way	
	10-shot \uparrow	20-shot \uparrow	10-shot \uparrow	20-shot \uparrow
<i>Supervised Learning Only</i>				
○ DGCNN [53]	31.6 \pm 2.8	40.8 \pm 4.6	19.9 \pm 2.1	16.9 \pm 1.5
● Transformer [51]	87.8 \pm 5.2	93.3 \pm 4.3	84.6 \pm 5.5	89.4 \pm 6.3
★ MAMBA3D	92.6 \pm 3.7	96.9 \pm 2.4	88.1 \pm 5.3	93.1 \pm 3.6
<i>with Self-supervised Pre-training</i>				
○ DGCNN+OcCo	90.6 \pm 2.8	92.5 \pm 1.9	82.9 \pm 1.3	86.5 \pm 2.2
● OcCo [52]	94.0 \pm 3.6	95.9 \pm 2.7	89.4 \pm 5.1	92.4 \pm 4.6
★ PointMamba [26]	95.0 \pm 2.3	97.3 \pm 1.8	91.4 \pm 4.4	92.8 \pm 4.0
● MaskPoint [27]	95.0 \pm 3.7	97.2 \pm 1.7	91.4 \pm 4.0	93.4 \pm 3.5
● Point-BERT [62]	94.6 \pm 3.1	96.3 \pm 2.7	91.0 \pm 5.4	92.7 \pm 5.1
★ MAMBA3D+P-B	95.8 \pm 2.7	97.9 \pm 1.4	91.3 \pm 4.7	94.5 \pm 3.3
● Point-MAE [34]	96.3 \pm 2.5	97.8 \pm 1.8	92.6 \pm 4.1	95.0 \pm 3.0
★ MAMBA3D+P-M	96.4 \pm 2.2	98.2 \pm 1.2	92.4 \pm 4.1	95.2 \pm 2.9

Results. Table 1 reports the results. When trained from scratch, MAMBA3D achieved 91.81% overall accuracy (OA) on the most difficult variant PB_T50_RS of ScanObjectNN, and 92.64% after voting, surpassing SoTA model DeLa’s 90.4% [5], achieving new SoTA for models trained from scratch. Compared to Transformer [51], MAMBA3D gains an OA increase of +15.40%, with only 76% parameters and 81% FLOPs. Notably, MAMBA3D surpasses two concurrent works PointMamba [26] and PCM [66] by +10.16% and +4.54%, respectively. On the ModelNet40 dataset, MAMBA3D is +2.7% higher than Transformer. Our model surpasses PCM with less than half the parameters (16.9M vs. 34.2M), and only 8.7% FLOPs (3.9G vs. 45.0G).

After pre-training, our proposed MAMBA3D consistently outperforms Transformer-based models. With the Point-BERT [62] strategy, MAMBA3D surpasses Point-BERT by +7.04% on ScanObjectNN and +1.2% on ModelNet40, also outperforming hierarchical Transformer model Point-M2AE [65] by +1.15% and +0.4% on these two datasets. When using the Point-MAE [34] strategy, MAMBA3D achieves 95.1% on ModelNet40, setting new SoTA for single-modal pre-trained models. On the ScanObjectNN dataset, MAMBA3D outperforms Transformer with OcCo [52] by +14.55%, and Point-MAE by +8.16%. Besides, we gained an increase of +8.47% compared to PointMamba [26] with the same pre-training strategy. Overall, these results highlight MAMBA3D’s superiority over existing dedicated architectures and Transformer- or Mamba-based models, achieving multiple SoTA, demonstrating its strength across various settings.

4.2.2 Few-shot Learning. We conduct few-shot classification experiments following previous work [47], to further validate few-shot learning ability of MAMBA3D.

Settings. We use ModelNet40 dataset [57] with an n -way, m -shot setting, where n is the number of classes randomly sampled from the dataset, and m denotes the number of samples randomly drawn from each class. We train the model with only the sampled

Table 3: Part segmentation on ShapeNetPart dataset. The class mIoU ($mIoU_C$) and the instance mIoU ($mIoU_I$) are reported, with model parameters #P (M) and FLOPs #F (G).

Method	$mIoU_C$ (%) \uparrow	$mIoU_I$ (%) \uparrow	#P \downarrow	#F \downarrow
<i>Supervised Learning Only</i>				
○ PointNet [36]	80.4	83.7	3.6	4.9
○ PointNet++ [37]	81.9	85.1	1.0	4.9
○ DGCNN [53]	82.3	85.2	1.3	12.4
● Transformer [51]	83.4	85.1	27.1	15.5
★ MAMBA3D	83.7	85.7	23.0	11.8
<i>with Self-supervised Pre-training</i>				
● OcCo [52]	83.4	84.7	27.1	-
○ PointContrast [58]	-	85.1	37.9	-
○ CrossPoint [1]	-	85.5	-	-
● Point-BERT [62]	84.1	85.6	27.1	10.6
★ MAMBA3D+P-B	84.1	85.7	21.9	9.5
● Point-MAE [34]	84.2	86.1	27.1	15.5
★ PointMamba [26]	84.4	86.0	17.4	14.3
★ MAMBA3D+P-M	83.6	85.6	23.0	11.8

$n \times m$ samples. During testing, we randomly select 20 novel objects for each of the n classes to serve as test data. We experiment with $n \in \{5, 10\}$ and $m \in \{10, 20\}$. For each setting, we report the mean accuracy and standard deviation of 10 independent experiments.

Results. Table 2 reports the comparison results. When trained from scratch, Both MAMBA3D and Transformer significantly surpass DGCNN [53] by a large margin. MAMBA3D outperforms Transformer [51] with overall accuracy (OA) improvements of +4.8%, +3.6%, +3.5%, and +3.7% across four settings, respectively, with also smaller deviations and fewer FLOPs. Under the Point-BERT strategy, MAMBA3D outperformed Point-BERT [62] by +1.2%, +1.6%, +0.3%, and +1.8%, respectively, and with smaller deviations. Similarly, with the Point-MAE [34] strategy, MAMBA3D outperforms Point-MAE on three out of four settings, and surpasses PointMamba [26] on all settings. These few-shot experiments demonstrate MAMBA3D’s adeptness at learning semantic information and its efficient knowledge transfer ability to downstream tasks, even with limited data.

4.2.3 Part Segmentation. We conducted part segmentation on the ShapeNetPart dataset [60] to predict more fine-grained class labels for every point.

Settings. ShapeNetPart dataset comprises ~16K objects across 16 categories. We use a segmentation head similar to Point-BERT [62]-utilizing PointNet++ [37] in feature propagation, along with a similar feature extraction strategy—employing features from the 4th, 8th, and 12th encoder layers. We use input point cloud $N=2048$ without normal, and employ cross-entropy as the loss function.

Results. Table 3 reports the comparison results in terms of the average instance IoU ($mIoU_I$) and average category IoU ($mIoU_C$). With supervised training alone, MAMBA3D surpasses Transformer by +0.3% in $mIoU_C$ and +0.6% in $mIoU_I$. With the Point-BERT

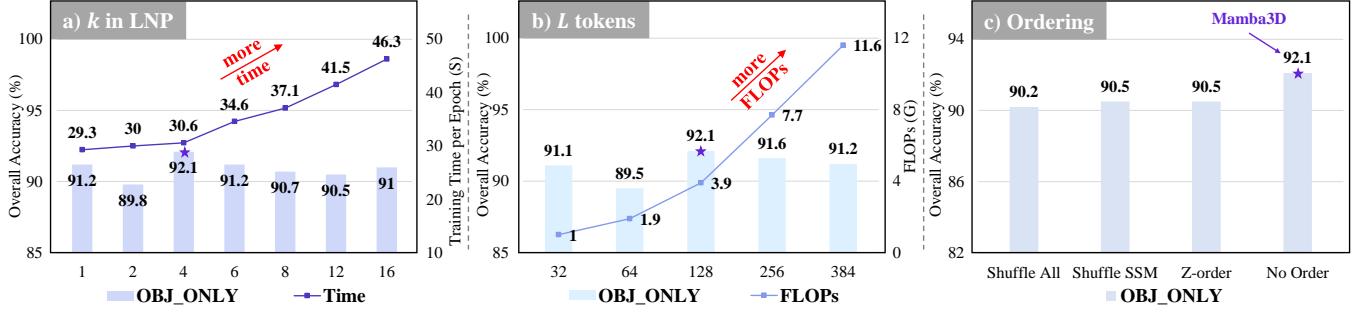


Figure 4: Ablation on (a) the parameter k in the LNP block, (b) input patch sequence length L , and (c) ordering strategy. The overall accuracy (%), training time/epoch (s) and FLOPs (G) are reported.

Table 4: Ablation on model architecture.

Method	OBJ_ONLY (%) \uparrow	Params (M) \downarrow	FLOPs (G) \downarrow
★ Full	92.1	16.9	3.9
w/o LNP	90.9 -1.2	13.3	3.4
w/o bi-SSM	89.8 -2.3	4.4	2.5
★ tri-SSM	91.0 -1.1	17.9	3.9
★ one-SSM	90.9 -1.2	16.0	3.9
● LNP + Attn	90.9 -1.2	25.7	5.4
★ Token Flip	90.7 -1.4	16.9	3.9

strategy, MAMBA3D achieves +0.1% higher mIoU₁ compared to Point-BERT. While MAMBA3D yields slightly lower results than Point-MAE, it employs 17.8% fewer parameters and 31.4% fewer FLOPs. The segmentation experiments further demonstrate the effectiveness and efficiency of MAMBA3D.

4.3 Ablation Study

We conduct ablation studies on model structure and also investigate the effect of ordering strategies. We report the results of training from scratch on the ScanObjectNN (OBJ_ONLY) dataset.

4.3.1 Architecture Ablation. Results of ablation on architecture are shown in Table 4. Removing the LNP block (w/o LNP) and the bi-SSM block (w/o bi-SSM) separately results in a 1.2% and 2.3% degradation in overall accuracy (OA), respectively. To further validate the effect of the bi-SSM block, we design four variants. Firstly, we replace it with a self-attention [51] layer (LNP+Attn), which leads to a 1.2% reduction in OA. When using only a unidirectional SSM (one-SSM), the OA decreases by 1.2%. Exploring token flip as an alternative to the channel flip (Token Flip) results in a 1.4% OA degradation, and directly adding a token flip in bi-SSM block (tri-SSM) leads to a 1.1% drop. These results demonstrate the effectiveness of the C-SSM block for unordered points.

4.3.2 K-norm Ablation. Results of ablation on K-norm are shown in Table 5. Removing K-norm entirely (w/o K-norm) decreases OA by 2.9%. We extensively verify the effectiveness of the K-norm operator defined in Eq. (5). Dropping the concatenated F_C (w/o concat.) lowered OA to 88.6%, and removing F_C completely (w/o F_C) decreases OA by 3.3%. Without linear transformation (w/o linear), OA decreases by 1.6%, and using distinct linear transformations

Table 5: Ablation on K-norm and K-pooling.

Method	OBJ_ONLY (%) \uparrow	Params (M) \downarrow	FLOPs (G) \downarrow
K-norm	92.1	16.93	3.86
w/o rela. dist.	91.6 -0.5	16.93	3.86
w/ K linear	90.9 -1.2	16.98	3.86
w/o linear	90.5 -1.6	16.91	3.86
w/o K-norm	89.2 -2.9	21.25	5.98
w/o F_C	88.8 -3.3	15.14	3.63
w/o concat.	88.6 -3.5	15.14	3.63
K-pooling	92.1	16.93	3.86
w/ Maxpool	91.4 -0.7	16.93	3.86
w/o K-pool	89.8 -2.3	17.72	4.47
w/ Max + Avgpool	89.7 -2.4	16.93	3.86
w/ Avgpool	89.3 -2.8	16.93	3.86

for K neighbors (K linear) leads to a 1.2% drop, underscoring K-norm's simplicity and efficacy. Even without the centralizing F_C , the model achieved 91.6% OA. These results highlight the K-norm's role in effectively transmitting local information and improving local geometric capture.

4.3.3 K-pooling Ablation. More detailed ablation results for K-pooling are shown in Table 5. Replacing K-pooling with one MLP (w/o K-pool) leads to an increase of +0.8M in parameters and +0.61G in FLOPs, while the OA decreases by 2.3%, highlighting the simplicity and effectiveness of K-pooling. When K-pooling is substituted with Avgpooling, Maxpooling, or Max+Avgpooling, the OA is reduced by 2.8%, 0.7%, and 2.4%, respectively, indicating the efficacy of the simple K-pooling operator in aggregating local features.

4.3.4 Parameters and Ordering. We also analyze the model's performance under various model parameters, as depicted in Fig. 4(a-b). In Fig. 4(a), we adjust the k in LNP to change the size of the local patch graph. Results show that as the local neighborhood increases, so does the model training time, with the overall accuracy (OA) reaching its peak at $k=4$. Results in Fig. 4(b) indicate that as the token length L increases, so do the FLOPs, with the OA peaking at 92.1% when $L=128$. Lastly, in Fig. 4(c), we investigate ordering strategies and find that MAMBA3D performs optimally without any ordering strategy applied. This suggests that MAMBA3D effectively captures the semantic features from unordered points.

5 Conclusion

We introduce MAMBA3D, a novel SSM-based architecture for point cloud learning. MAMBA3D surpasses Transformer-based models across various tasks while maintaining linear complexity. Specifically, our LNP block, comprising K-norm and K-pooling, facilitates the explicit local feature injection. Also, we propose C-SSM to adapt the SSM for better handling unordered points. Through extensive experimentation and validation, MAMBA3D achieves multiple SoTA across multiple tasks with only linear complexity. We aspire for MAMBA3D to advance the field of large point cloud models.

Discussion. There are still limitations to this work. Pre-training bonus is not as robust as the Transformer, possibly due to unsuitable masked point modeling for recurrent models like Mamba. We will explore tailored pre-training strategies and scale up the model to optimize its linear complexity advantage in future research.

Acknowledgments

We sincerely thank Qiao Yu for proofreading. This work was supported by the China National Natural Science Foundation No. 62202182. The computation is completed in the HPC Platform of Huazhong University of Science and Technology.

References

- [1] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. 2022. CrossPoint: Self-Supervised Cross-Modal Contrastive Learning for 3D Point Cloud Understanding. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*.
- [2] Matan Atzmon, Haggai Maron, and Yaron Lipman. 2018. Point Convolutional Neural Networks by Extension Operators. *arXiv preprint arXiv:1803.10091* (2018).
- [3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).
- [4] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR abs/1512.03012* (2015). arXiv:1512.03012
- [5] Binjie Chen, Yunzhou Xia, Yu Zang, Cheng Wang, and Jonathan Li. 2023. Decoupled Local Aggregation for Point Cloud Learning. *arXiv preprint arXiv:2308.16532* (2023).
- [6] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. 2024. PointGPT: Auto-regressively Generative Pre-training from Point Clouds. *Adv. Neural Inform. Process. Syst. (NeurIPS) 36* (2024).
- [7] Sihlin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. 2021. PRA-Net: Point Relation-Aware Network for 3D Point Cloud Analysis. *IEEE Trans. Image Process. (TIP) 30* (2021), 4436–4448.
- [8] Xin Deng, WenYu Zhang, Qing Ding, and XinMing Zhang. 2023. PointVector: A Vector Representation in Point Cloud Analysis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 9455–9465.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.
- [10] Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. 2023. Autoencoders as Cross-Modal Teachers: Can Pretrained 2D Image Transformers Help 3D Representation Learning?. In *Int. Conf. Learn. Represent. (ICLR)*.
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Int. Conf. Learn. Represent. (ICLR)*.
- [12] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*.
- [13] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. 2022. It's Raw! Audio Generation with State-Space Models. In *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 7616–7633.
- [14] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. 2021. Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline. In *Proc. Int. Conf. Mach. Learn. (ICML) (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 3809–3820.
- [15] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752* (2023).
- [16] Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv preprint arXiv:2111.00396* (2021).
- [17] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. *Adv. Neural Inform. Process. Syst. (NeurIPS) 34* (2021), 572–585.
- [18] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. 2021. PCT: Point cloud transformer. *Comput. Vis. Media 7*, 2 (2021), 187–199.
- [19] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 43*, 12 (2020), 4338–4364.
- [20] Abdulllah Hamdi, Silvio Giancola, and Bernard Ghanem. 2021. MVTN: Multi-View Transformation Network for 3D Shape Recognition. In *Int. Conf. Comput. Vis. (ICCV)*. IEEE, 1–11.
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. IEEE Computer Society, 770–778.
- [22] Rudolph Emil Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. (1960).
- [23] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. 2022. Stratified Transformer for 3D Point Cloud Segmentation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 8500–8509.
- [24] Loic Landrieu and Martin Simonovsky. 2018. Large-Scale Point Cloud Semantic Segmentation With Superpoint Graphs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 4558–4567.
- [25] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhuan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Adv. Neural Inform. Process. Syst. (NeurIPS) 32*. 828–838.
- [26] Dingkang Liang, Xin Zhou, Xinyu Wang, Xingkui Zhu, Wei Xu, Zhikang Zou, Xiaoqing Ye, and Xiang Bai. 2024. PointMamba: A Simple State Space Model for Point Cloud Analysis. *arXiv preprint arXiv:2402.10739* (2024).
- [27] Haotian Liu, Mu Cai, and Yong Jae Lee. 2022. Masked Discrimination for Self-Supervised Learning on Point Clouds. In *Eur. Conf. Comput. Vis. (ECCV)*.
- [28] Yang Liu, Chen Chen, Can Wang, Xulin King, and Mengyuan Liu. 2023. Regress Before Construct: Regress Autoencoder for Point Cloud Self-supervised Learning. In *ACM Int. Conf. Multimedia (ACM MM)*. 1738–1749.
- [29] Yahui Liu, Bin Tian, Yisheng Lv, Lingxi Li, and Fei-Yue Wang. 2023. Point Cloud Classification Using Content-Based Transformer via Clustering in Feature Space. *IEEE/CAA Journal of Automatica Sinica* (2023).
- [30] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Int. Conf. Learn. Represent. (ICLR)*.
- [31] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. 2022. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. In *Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net.
- [32] Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. 2022. S4ND: Modeling Images and Videos as Multidimensional Signals with State Spaces. *Adv. Neural Inform. Process. Syst. (NeurIPS) 35* (2022), 2846–2861.
- [33] Zirui Pan, Mengbai Xiao, Xu Han, Dongxiao Yu, Guanghui Zhang, and Yao Liu. 2024. patchDPCC: A Patchwise Deep Compression Framework for Dynamic Point Clouds. In *AAAI Conf. Artif. Intell. (AAAI)*. Vol. 38. 4406–4414.
- [34] Yatian Pang, Wenxiang Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. 2022. Masked Autoencoders for Point Cloud Self-supervised Learning. In *Eur. Conf. Comput. Vis. (ECCV)*.
- [35] Jinyoung Park, Sanghyeon Lee, Sihyeon Kim, Yunyang Xiong, and Hyunwoo J. Kim. 2023. Self-Positioning Point-Based Transformer for Point Cloud Understanding. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 21814–21823.
- [36] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 77–85.
- [37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Adv. Neural Inform. Process. Syst. (NeurIPS) 30*. 5099–5108.
- [38] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. 2021. Offboard 3D Object Detection From Point Cloud Sequences. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 6134–6144.
- [39] Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. 2023. Contrast with Reconstruct: Contrastive 3D Representation Learning Guided by Generative Pretraining. In *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 28223–28243.
- [40] Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. 2024. ShapeLLM: Universal 3D Object Understanding

- for Embodied Interaction. *arXiv preprint arXiv:2402.17766* (2024).
- [41] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*.
- [42] Shi Qiu, Saeed Anwar, and Nick Barnes. 2021. Dense-Resolution Network for Point Cloud Classification and Segmentation. In *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. 3812–3821.
- [43] Shi Qiu, Saeed Anwar, and Nick Barnes. 2022. Geometric Back-Projection Network for Point Cloud Classification. *IEEE Trans. Multimedia (TMM)* 24 (2022), 1943–1955.
- [44] Haoxi Ran, Jun Liu, and Chengjie Wang. 2022. Surface Representation for Point Clouds. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 18942–18952.
- [45] Jason Tyler Rolfe. 2016. Discrete Variational Autoencoders. *arXiv preprint arXiv:1609.02200* (2016).
- [46] Radu Bogdan Rusu and Steve Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 1–4.
- [47] Charu Sharma and Manohar Kaul. 2020. Self-Supervised Few-Shot Learning on Point Clouds. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*.
- [48] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 770–779.
- [49] Yuan Tang, Xianzhi Li, Jinfeng Xu, Qiao Yu, Long Hu, Yixue Hao, and Min Chen. 2023. Point-LGMask: Local and Global Contexts Embedding for Point Cloud Pre-training with Multi-Ratio Masking. *IEEE Transactions on Multimedia* (2023).
- [50] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 1588–1597.
- [51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*. 5998–6008.
- [52] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. 2021. Unsupervised point cloud pre-training via occlusion completion. In *Int. Conf. Comput. Vis. (ICCV)*. 9782–9792.
- [53] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* 38, 5 (2019), 146:1–146:12.
- [54] Ziyi Wang, Xumin Yu, Yongming Rao, Jie Zhou, and Jiwen Lu. 2022. P2P: Tuning Pre-trained Image Models for Point Cloud Analysis with Point-to-Pixel Prompting. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*.
- [55] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2023. Point transformer v3: Simpler, faster, stronger. *arXiv preprint arXiv:2312.10035* (2023).
- [56] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. 2022. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems* 35 (2022), 33330–33342.
- [57] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 1912–1920.
- [58] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. 2020. PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Eur. Conf. Comput. Vis. (ECCV) (Lecture Notes in Computer Science, Vol. 12348)*. Springer, 574–591.
- [59] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. 2023. ULIP-2: Towards Scalable Multimodal Pre-training for 3D Understanding. *arXiv preprint arXiv:2305.08275* (2023).
- [60] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.* 35, 6 (2016), 1–12.
- [61] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiaoshi Feng, and Shuicheng Yan. 2022. MetaFormer Is Actually What You Need for Vision. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 10819–10829.
- [62] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*.
- [63] Karim Abou Zeid, Jonas Schult, Alexander Hermans, and Bastian Leibe. 2023. Point2Vec for Self-Supervised Representation Learning on Point Clouds. *arXiv preprint arXiv:2303.16570* (2023).
- [64] Yaohua Zha, Jinpeng Wang, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. 2023. Instance-aware Dynamic Prompt Tuning for Pre-trained Point Cloud Models. In *Int. Conf. Comput. Vis. (ICCV)*. 14161–14170.
- [65] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. 2022. Point-M2AE: Multi-scale Masked Autoencoders for Hierarchical Point Cloud Pre-training. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*.
- [66] Tao Zhang, Xiangtai Li, Haobo Yuan, Shunping Ji, and Shuicheng Yan. 2024. Point Could Mamba: Point Cloud Learning via State Space Model. *arXiv preprint arXiv:2403.00762* (2024).
- [67] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. 2019. PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 5565–5573.
- [68] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. 2021. Point Transformer. In *Int. Conf. Comput. Vis. (ICCV)*. IEEE, 16239–16248.
- [69] Xin Zhou, Dingkang Liang, Wei Xu, Xingkui Zhu, Yihai Xu, Zhikang Zou, and Xiang Bai. 2024. Dynamic Adapter Meets Prompt Tuning: Parameter-Efficient Transfer Learning for Point Cloud Analysis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. (CVPR)*. 14707–14717.
- [70] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model. *arXiv preprint arXiv:2401.09417* (2024).