

MoST: Efficient Monarch Sparse Tuning for 3D Representation Learning

Xu Han Yuan Tang Jinfeng Xu Xianzhi Li[†]

Huazhong University of Science and Technology

{xhanxu, yuan.tang, jinfengx, xzli}@hust.edu.cn [†]Corresponding author.

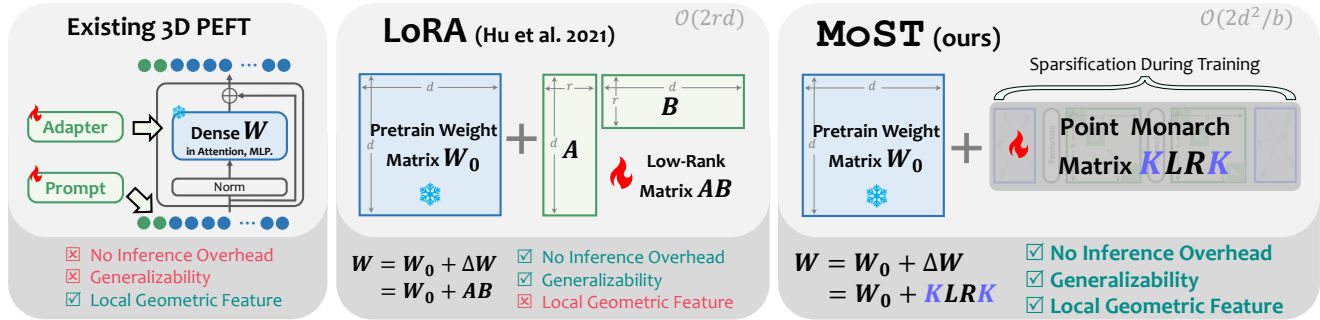


Figure 1. Existing 3D parameter-efficient fine-tuning (PEFT) methods rely on additional adapters or prompts, which, while using point cloud priors, introduce inference overhead and lack generalization. Reparameterization-based PEFT methods like LoRA[23], though free of the above issues, overlook point cloud characteristics. MoST combines the best of both worlds by reparameterizing dense update weight matrices with tailored sparse Point Monarch matrices, preserving local geometry, avoiding inference overhead, and remaining generalizable.

Abstract

We introduce *Monarch Sparse Tuning (MoST)*, the first reparameterization-based parameter-efficient fine-tuning (PEFT) method tailored for 3D representation learning. Unlike existing adapter-based and prompt-tuning 3D PEFT methods, MoST introduces no additional inference overhead and is compatible with many 3D representation learning backbones. At its core, we present a new family of structured matrices for 3D point clouds, *Point Monarch*, which can capture local geometric features of irregular points while offering high expressiveness. MoST reparameterizes the dense update weight matrices as our sparse *Point Monarch* matrices, significantly reducing parameters while retaining strong performance. Experiments on various backbones show that MoST is simple, effective, and highly generalizable. It captures local features in point clouds, achieving state-of-the-art results on multiple benchmarks, e.g., 97.5% acc. on ScanObjectNN (PB_50_RS) and 96.2% on ModelNet40 classification, while it can also combine with other matrix decompositions (e.g., Low-rank, Kronecker) to further reduce parameters.

1. Introduction

3D point cloud analysis finds extensive applications in autonomous driving [37, 44], embodied AI [67], and virtual reality [18]. In recent years, learning-based point cloud

analysis has advanced rapidly, achieving leading performance across various tasks. Pioneered by PointNet [35], most point cloud learning models focus on designing intricate models to learn geometric or semantic features from unstructured points [29, 36, 40, 50, 54]. Inspired by the success of large-scale pretraining in language and image modeling [2, 9, 21], recent approaches in point cloud learning [4, 20, 33, 34, 58, 61] leverage pretrained backbones like Transformer [53], Mamba [16], and hierarchical architectures [42] for 3D representation learning. This pretrain-finetune paradigm leads to more robust point cloud features, boosting performance on downstream tasks [56].

A problem arises when applying pretrained models to downstream tasks: the entire model needs to be retrained on downstream datasets, requiring substantial computation and memory—impractical for many tasks. To address this, recent efforts on Parameter-Efficient Fine-Tuning (PEFT) [30] aim to achieve strong performance with minimal parameter updates. PEFT methods fall into three categories: additive (e.g., Adapters [22], Prompt Tuning [27]), selective (e.g., BitFit [59]), and reparameterization-based (e.g., LoRA [23]). Additive methods introduce new model components or input tokens, training only these additional parameters. Selective methods fine-tune a specific subset of the model. Reparameterization-based methods replace dense update weight matrices with sparse approximations, like the low-rank approximation used in LoRA. Given that most models

contain numerous dense layers, such replacements offer a superior balance between generalization and efficiency.

Existing PEFT methods primarily target language and image modeling [10, 30], often falling short when applied to 3D point clouds. The field of 3D PEFT remains underexplored, with most efforts focusing on additive methods that use custom adapters or prompts tailored for Transformer [48, 60, 63, 66]. While these methods significantly reduce the fine-tuning parameters and effectively incorporate point cloud priors, such as local features, they face two key limitations: (1) Limited generalization and flexibility: adapters or prompts designed for Transformer [53] do not generalize well to other backbones like Mamba [16] or hierarchical models [42], and adjusting parameter sizes is difficult, hindering scalability. (2) They introduce additional inference overhead and often fail to match the performance of full fine-tuning. Intuitively, a reparameterization-based PEFT method offers a solution by preserving the model architecture and avoiding inference overhead, as it simply reparameterizes dense weight matrices with sparse ones during training. However, in practice, methods like LoRA [23], which rely on low intrinsic rank, tend to capture global information but struggle with local feature learning. As shown in Fig. 2, this limitation severely impacts 3D PEFT performance.

To fill this gap, we propose Monarch Sparse Tuning (MoST), the first reparameterization-based 3D PEFT method. At its core, we introduce Point Monarch, a novel structured matrix family tailored for 3D point clouds, extending Monarch [7] to unstructured points. Point Monarch captures local geometric features with linear transformations, and features a block-wise structure that aligns well with patch-based point cloud learning, outperforming low-rank matrices. MoST replaces dense update weight matrices with Point Monarch during training, supporting various backbones without adding inference overhead. MoST captures local features effectively and is highly expressive, enabling it to match or exceed full fine-tuning across multiple tasks. Additionally, MoST allows users to balance efficiency and performance based on task needs, and supports further parameter reduction via matrix decompositions, such as Low-rank and Kronecker [52]. Finally, we design a parameter-free multi-layer feature fusion strategy, to boost knowledge transfer from the backbone to the task header, avoiding bottlenecks.

We validate MoST across diverse settings, including masked point, multimodal, and large-scale pretraining, using various backbones such as Transformers, Mamba, and hierarchical architectures. Extensive experiments show that MoST consistently outperforms existing PEFT methods in object- and scene-level classification and segmentation tasks, even surpassing full fine-tuning in almost all classification tasks while tuning just 3.6% of parameters. With MoST, Point-MAE [34] and I2P-MAE [62] achieve accuracy gains of 7.74% and 3.16%, respectively, over full fine-tuning on

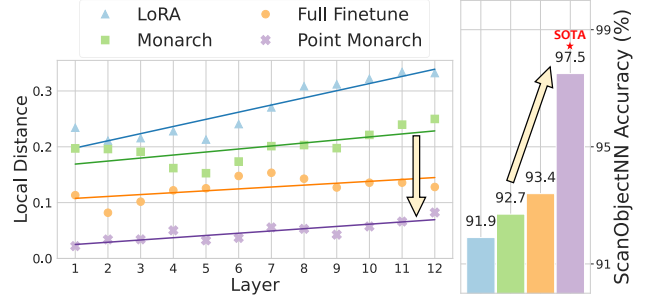


Figure 2. The average l_2 -distance of features between KNN centers and neighbors after applying different structured matrices (left). We observe this local feature distance correlates with classification acc. on PB_50_RS [51] (right). LoRA and Monarch [7] lead to higher distances, Point Monarch exhibits the lowest one by smoothing local geometric features, achieving 97.5% acc. with PointGPT [4].

the ScanObjectNN [51] (PB_50_RS). Mamba3D [20] and PointGPT [4] achieve 95.2% and **96.2%** accuracy on ModelNet40 [55], and 93.3% and **97.5%** on PB_50_RS. MoST fine-tuned ReCon [38] surpasses the state-of-the-art 3D PEFT method by 0.2% class mIoU on ShapeNetPart [57] part segmentation and 1.0% mIoU on S3DIS [1] scene segmentation.

In summary, our main contributions are as follows.

- We introduce Point Monarch, a novel structured matrix family tailored for 3D point clouds that captures both global and local features while maintaining sparsity.
- We propose MoST, the first reparameterization-based 3D PEFT method, which generalizes well across different backbones without adding inference overhead.
- Extensive results across various models demonstrate MoST’s generalization and effectiveness, achieving state-of-the-art 3D PEFT on object- and scene-level tasks.

2. Related Work

2.1. 3D Representation Learning

3D representation learning [56] aims to extract robust point cloud features through pretraining on large datasets, boosting performance in downstream tasks. Inspired by Masked Language Modeling [9] and Masked Image Modeling [21], many point cloud pretraining methods mask parts of the point cloud and train models to reconstruct these masked regions [56]. Notable methods like Point-BERT [58] and Point-MAE [34] leverage this to learn effective representations, excelling in tasks such as classification and segmentation. Recently, with the increasing availability of 3D-text data pairs, multimodal pretraining methods like ACT [11] and ReCon [38, 39] have shown superior effectiveness. However, fine-tuning the entire pretrained models for downstream tasks is resource-intensive, calling for effective 3D Parameter-Efficient Fine-Tuning (PEFT) [30]. Here, we introduce the first reparameterization-based 3D PEFT method, specifically designed for irregular 3D point clouds.

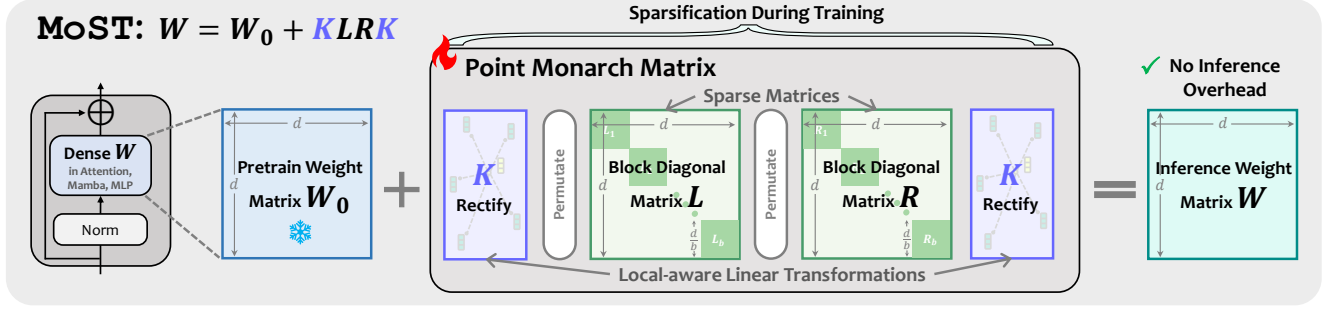


Figure 3. **Illustration of Monarch Sparse Tuning.** During training, MoST reparameterizes dense update weight matrices using our sparse and expressive Point Monarch matrices $\mathbf{KLRK}$, which capture local geometric features of points through simple linear transformations.

2.2. 3D Parameter-Efficient Fine-Tuning

Current 3D PEFT primarily focuses on adapter [22] and prompt tuning [27]. Pioneers like IDPT [60] and DAPT [66] explore this approach: IDPT employs DGCNN [54] as an adapter to generate prompts for Transformers, while DAPT uses a dynamic adapter that combines adapter and prompt tuning. Both methods significantly reduce trainable parameters, leveraging point cloud features to enhance performance. Recently, PPT [63] demonstrates the effectiveness of fine-tuning position encodings, and PointGST [32] proposes an adapter to extract spectral-domain features for efficient 3D PEFT. However, these adapter-based and prompt tuning methods [12, 13, 47–49] introduce inference overhead and are specifically designed for Transformers [53], limiting adaptability to other architectures like Mamba [16] and U-Net [42]. In contrast, our MoST overcomes these limitations from the perspective of reparameterization, avoiding inference overhead while maintaining high generalizability.

2.3. Structured Matrices

Dense matrices exhibit quadratic complexity, whereas structured matrices offer sub-quadratic parameter counts and runtime [41], making them ideal for replacing dense layers in large models to reduce computation. Common structured matrices include Low-rank, Kronecker, and Toeplitz matrices [46] (or known as convolutions), as well as various fast transforms such as Fourier and sine/cosine. Notably, inspired by the divide-and-conquer scheme from De Sa et al. [8], Dao et al. [6] introduce Butterfly matrices, which are highly expressive but inefficient on modern GPUs. Monarch matrices [7] then extend Butterfly by leveraging optimized batch-matrix-multiply to accelerate factorization. However, these structured matrices often struggle to capture local point cloud features, limiting their effectiveness in 3D PEFT. In contrast, our Point Monarch captures local geometric features using *two simple KNN-wise local token linear transformations, seamlessly integrating with efficient batch matrix multiplication* while maintaining a block-diagonal structure suited for patch-based point cloud learning.

3. Method

During training, MoST (Sec. 3.3) reparameterizes dense update weight matrices into sparse Point Monarch (Sec. 3.2), extending Monarch (Sec. 3.1) to point cloud learning, capturing local features while maintaining high expressiveness. We also propose a feature fusion strategy (Sec. 3.4) to enhance the backbone-header alignment without learning.

3.1. Preliminaries: Monarch Matrices

Dao et al. [7] introduce Monarch to enhance hardware efficiency over Butterfly [6]. Following sparse matrix product factorization [8], a Monarch matrix $\mathbf{M}$ is defined as:

$$\mathbf{M} = \mathbf{P}\mathbf{L}\mathbf{P}^\top\mathbf{R}, \quad (1)$$

where $\mathbf{P}$ is a permutation from row-major to column-major, commonly used in CNNs like ShuffleNet [64]. $\mathbf{L}$ and $\mathbf{R}$ are sparse block-diagonal matrices of size d with b blocks, where each block is of size d/b . Typically, with $b \approx \sqrt{d}$, Monarch’s total parameter count is $2d^2/b \ll d^2$.

As shown in Fig. 2, we observe that the average l_2 -distance of features between KNN centers and neighbors correlates with classification accuracy on ScanObjectNN [51], with lower distances indicating higher accuracy. We assume that this local feature distance reflects the model’s ability to capture local features. Both LoRA and Monarch show higher distances than full fine-tuning, leading to lower accuracy. Despite Monarch’s higher expressiveness compared to LoRA [23], it still struggles to capture local geometric features in point clouds, limiting its performance in 3D PEFT tasks. Inspired by this, we design Point Monarch to reduce the local feature distance and optimize performance.

3.2. Point Monarch

Following the divide-and-conquer factorization scheme [8], we introduce Point Monarch, a structured matrix family tailored for 3D point clouds, as shown in Fig. 3, defined as:

$$\text{Point Monarch} = \mathbf{K}\mathbf{P}\mathbf{L}\mathbf{P}^\top\mathbf{R}\mathbf{K}, \quad (2)$$

where $\mathbf{K}$, representing K -Rectify, facilitates information exchange among tokens within geometric (xyz) neighborhoods,

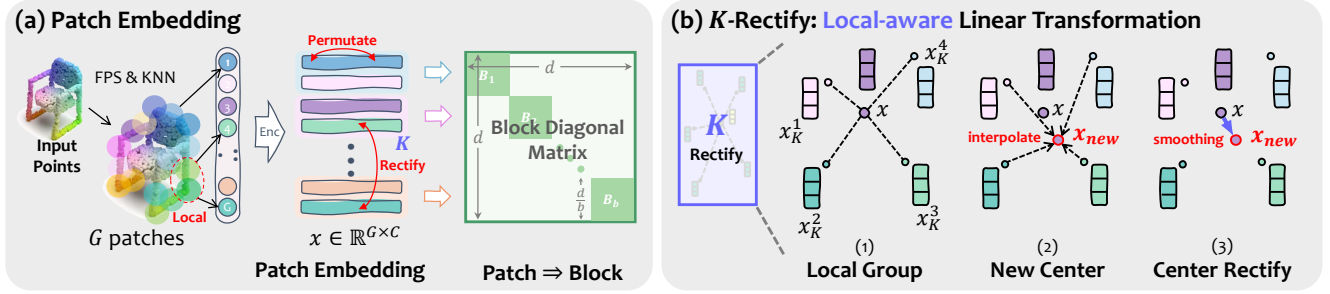


Figure 4. **Illustration of Patch Embedding and K -Rectify.** (a) Our Point Monarch performs channel-wise permutation and token-wise local rectification, and its block-wise structure aligns with patch-based point cloud representation learning. (b) K -Rectify groups local features based on xyz coordinates and interpolates a new center feature, then facilitating the center feature rectification.

differing from channel information fusion with permutation **P**. **L** and **R** are block-diagonal matrices of size $d \times d$, with b blocks of $\mathbf{L}_i/\mathbf{R}_i \in \mathbb{R}^{\frac{d}{b} \times \frac{d}{b}}$ on the diagonal:

$$\mathbf{L} = \text{diag}(\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_b), \quad (3)$$

$$\mathbf{R} = \text{diag}(\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_b). \quad (4)$$

Point Monarch supports efficient batch matrix multiplication, making it hardware-friendly. It balances efficiency, expressiveness, and local feature capturing, with a parameter count of $2d^2/b \ll d^2$ where typically $b \approx \sqrt{d}$ to control sparsity. Replacing dense layers with sparse Point Monarch improves parameter efficiency, similar to model pruning [19], and actually can be seen as a handcrafted lottery ticket structure [14] tailored for point cloud processing. Here, **K** serves as a *KNN-wise local token linear transformation, integrating seamlessly with batch matrix multiplication while preserving the block-diagonal Monarch structure*. For simplicity, we abbreviate Point Monarch as **KLRK**.

K -Rectify. As shown in Fig. 4, **K** rectifies patch embeddings within local neighborhoods (Fig. 4a) through three steps (Fig. 4b): KNN local grouping, new center token interpolation, and rectification. For a token sequence $\mathbf{x} \in \mathbb{R}^{G \times C}$, where G is the sequence length, and C is the number of feature channels, **K** (K -Rectify) is defined as:

$$\mathbf{x}_K = \text{KNN}(\mathbf{p}, \mathbf{x}), \quad (5)$$

$$\mathbf{x}_{\text{new}} = \text{IDW}(\mathbf{x}_K), \quad (6)$$

$$\mathbf{K}\mathbf{x} = \mathbf{x} + \lambda\mathbf{x}_{\text{new}}. \quad (7)$$

Here, $\mathbf{x}$ contains embeddings for center points $\mathbf{p} \in \mathbb{R}^{G \times 3}$ across G patches. The local embedding groups $\mathbf{x}_K \in \mathbb{R}^{G \times K \times C}$ are obtained by KNN to find geometric neighbors of $\mathbf{p}$, and grouping the corresponding embeddings, with group size K . We compute new center embeddings $\mathbf{x}_{\text{new}} \in \mathbb{R}^{G \times C}$ using inverse distance weighting (IDW) of local groups $\mathbf{x}_K$. Finally, $\mathbf{x}_{\text{new}}$ refines the old $\mathbf{x}$ with a hyperparameter λ , resulting in the rectified token sequence $\mathbf{K}\mathbf{x}$.

The IDW is formulated as:

$$\text{IDW}(\mathbf{x}_K) = \sum_{i=1}^K \frac{\frac{1}{\|p - p_i\|^2}}{\sum_{j=1}^K \frac{1}{\|p - p_j\|^2}} \cdot x_K^i, \quad i = 1, \dots, K \quad (8)$$

where p is one of the center points $\mathbf{p}$, and p_i is one of p 's KNN geometric neighbors.

Matrix Form. Given the adjacency matrix $\mathbf{A} \in \mathbb{R}^{G \times G}$, where $\mathbf{A}_{ij} = 1$ if points i and j are KNN neighbors, otherwise 0, and the distance matrix $\mathbf{D} \in \mathbb{R}^{G \times G}$, where $\mathbf{D}_{ij}$ is the normalized inverse distance $1/d_{ij}$ between points i and j , we express **K** $\in \mathbb{R}^{G \times G}$ as a linear transformation matrix:

$$\mathbf{K} = \mathbf{I} + \lambda\mathbf{A} \odot \mathbf{D},$$

where $\odot$ is the Hadamard product, $\mathbf{I} \in \mathbb{R}^{G \times G}$ is the identity matrix. See Appendix for proof. Note that **K** is also sparse.

Key Insight. Local features in point clouds are best captured by fusing information within KNN neighborhoods [35, 36]. This fusion smooths local features while preserving their distinctiveness. In point cloud representation learning, it facilitates interactions among tokens in KNN neighborhoods, balancing local and global feature learning. While MLPs, Attention [53], or Mamba [16] can capture global features, our K -Rectify enhances local feature fusion through two simple KNN-wise linear transformations, smoothing local features while remaining hardware efficiency.

Expressiveness. De Sa et al. [8] and Dao et al. [6, 7] demonstrate that all structured matrices can be expressed as products of Butterfly or Monarch, achieving optimal memory and runtime complexity with polylogarithmic factors. Since Point Monarch extends Monarch to points, it inherits its high expressiveness, near-optimal runtime, and parameter efficiency. Furthermore, Point Monarch smooths local geometric features, making it more effective for point clouds.

3.3. Monarch Sparse Tuning

Monarch Sparse Tuning (**MoST**) replaces dense update weight matrices with Point Monarch, leveraging its sparsity for efficient reparameterization during training. **MoST**

Table 1. **PEFT Methods on Various Models.** We report the number of trainable parameters #P (M), ScanObjectNN PB_50_RS [51] (SO) accuracy (%) w/o voting, and ModelNet40 [55] (MN) accuracy (%) w/o voting. We compare different PEFT methods across $\diamond$ hierarchical architectures, $\circ$ Transformers, and $\star$ Mamba-based models for 3D understanding. 3D: designed for 3D PEFT. NO: no inference overhead.

Method	3D	NO	$\circ$ Point-MAE [34]		$\diamond$ I2P-MAE [62]		$\circ$ ReCon [38]		$\star$ Mamba3D [20]		$\circ$ PointGPT [4]	
			#P $\downarrow$	SO / MN $\uparrow$	#P $\downarrow$	SO / MN $\uparrow$	#P $\downarrow$	SO / MN $\uparrow$	#P $\downarrow$	SO / MN $\uparrow$	#P $\downarrow$	SO / MN $\uparrow$
Full FT	-	-	22.1	85.18 / 93.8	15.3	90.11 / 93.7	22.1	90.01 / 92.5	16.9	92.05 / 94.7	360.5	93.4 / 94.1
<i>Additive-based Parameter-Efficient Fine-Tuning</i>												
VPT [24]	$\times$	$\times$	0.4	81.09 / 92.54	0.3	76.30 / 89.10	0.4	84.04 / 92.59	0.4	80.60 / 91.13	1.1	91.60 / 92.0
IDPT [60]	$\checkmark$	$\times$	1.7	88.34 / 93.64	1.6	85.70 / 93.19	1.7	88.13 / 93.64	1.7	87.34 / 93.23	10.0	92.99 / 93.4
DAPT [66]	$\checkmark$	$\times$	1.1	88.27 / 92.99	0.8	89.04 / 93.56	1.1	89.31 / 93.27	1.1	88.55 / 92.87	4.2	93.02 / 94.2
PPT [63]	$\checkmark$	$\times$	1.1	89.00 / 93.68	1.6	87.54 / 92.99	1.1	89.52 / 93.76	1.1	87.61 / 92.87	2.8	94.34 / 94.2
PointGST [32]	$\checkmark$	$\times$	0.6	89.3 / 93.5	0.5	87.27 / 93.35	0.6	89.49 / 93.6	0.6	89.97 / 93.72	2.4	94.83 / 94.8
<i>Selective or Reparameterization-based Parameter-Efficient Fine-Tuning</i>												
BitFit [59]	$\times$	$\checkmark$	0.3	82.62 / 92.42	0.3	85.18 / 92.54	0.3	85.15 / 92.34	0.3	88.97 / 92.67	0.9	91.64 / 92.79
LoRA [23]	$\times$	$\checkmark$	0.9	82.76 / 92.50	0.8	84.00 / 92.59	0.9	85.70 / 92.87	0.9	87.16 / 92.42	2.4	91.92 / 92.95
MoST_{b=32}	$\checkmark$	$\checkmark$	0.8	91.95 / 94.04	0.6	91.22 / 94.73	0.8	92.02 / 94.29	0.8	91.50 / 94.00	2.5	97.19 / 95.95
MoST_{b=16}	$\checkmark$	$\checkmark$	1.3	92.71 / 94.49	0.8	92.23 / 95.06	1.3	92.85 / 94.69	1.2	92.85 / 94.65	4.4	97.26 / 96.11
MoST_{b=8}	$\checkmark$	$\checkmark$	2.3	92.92 / 94.77	1.4	93.27 / 95.14	2.3	93.55 / 95.06	2.1	93.30 / 95.18	8.0	97.50 / 96.23

reduces parameters while capturing local features, making it highly **generalizable** and **flexible**, ideal for 3D PEFT. During inference, the reparameterized matrices merge seamlessly with pretrained weights, maintaining the original structure and achieving *zero inference overhead*.

Generalizable. For any pretrained dense weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times d}$ and a dense update matrix $\Delta \mathbf{W} \in \mathbb{R}^{d \times d}$ fine-tuned for a downstream task, MoST is formulated as:

$$h = \mathbf{W}x = \mathbf{W}_0x + \Delta \mathbf{W}x, \quad (9)$$

$$= \mathbf{W}_0x + \mathbf{KLRK}x. \quad (10)$$

MoST reparameterizes the dense update matrix $\Delta \mathbf{W}$ (with d^2 parameters) into Point Monarch, reducing it to $2d^2/b$ parameters, where typically $b \approx \sqrt{d}$ controls sparsity. Similar to LoRA [23], we initialize $\mathbf{L} = 0$ and $\mathbf{R} = \mathcal{N}(0, \sigma^2)$, starting with $\Delta \mathbf{W} = 0$. This reparameterization generalizes well across models with dense layers, in contrast to current methods tailoring prompts or adapters for Transformers.

In practice, Point Monarch can handle non-square weights, specifically $\Delta \mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ where $d_{in} \neq d_{out}$, allowing adaptation to various models and tasks. Assuming $d_{in} < d_{out}$, we define $\mathbf{R} \in \mathbb{R}^{d_{in} \times d_{in}}$ and $\mathbf{L} \in \mathbb{R}^{d_{in} \times d_{out}}$. This implies that diagonal blocks can be rectangular matrices. Moreover, we can adjust the size of b based on model size or task complexity to control the parameter scaling.

Flexible. MoST is also highly flexible. Replacing dense layers directly with Point Monarch significantly reduces parameters during both training and inference, sparsifying the entire model rather than just the update weights:

$$\text{Sparse Training: } h = \mathbf{W}x = \mathbf{KLRK}x. \quad (11)$$

Moreover, MoST is orthogonal to common matrix decompositions such as Low-rank and Kronecker [52], allowing

further parameter reduction when combined. We explore some of these variants in Sec. 4.4.

3.4. Multi-layer Feature Fusion

Our parameter-free fusion strategy aggregates features from multiple layers, avoiding bottlenecks between the pretrained backbone and the task header, boosting downstream tasks, defined as:

$$\mathbf{x}_{out} = \sum_{i=1}^L 2^{i-1} \cdot \text{MixPool}(\mathbf{x}_i). \quad (12)$$

Here, $\mathbf{x}_i \in \mathbb{R}^{G \times C}$ ($i = 1, \dots, L$) represents one token sequence from L selected layers. For a 12-layer Transformer or Mamba model, we select $L = 3$ layers: the 4th, 8th, and 12th. Each sequence undergoes a mixture of MaxPooling and MeanPooling over its sequence length G , followed by a weighted sum 2^{i-1} ($i = 1, \dots, L$), yielding $\mathbf{x}_{out} \in \mathbb{R}^{1 \times C}$. Finally, we concatenate $\mathbf{x}_{out}$ with the class token from the last layer, which serves as input to the task header.

4. Experiments

4.1. 3D Representation Learning Backbones

We assess the generalization and effectiveness of MoST using five distinct point cloud pretraining model backbones:

Point-MAE [34]: A Transformer (22.1M) pretrained with masked point modeling (MPM) on ShapeNet [3] ($\sim 50K$).

I2P-MAE [62]: A U-Net-like hierarchical architecture (15.3M) with multimodal pretraining.

ReCon [38]: A Transformer (22.1M) with multimodal pretraining on ShapeNet [3].

Mamba3D [20]: A Mamba model (16.9M) pretrained with MPM on ShapeNet [3].

PointGPT [32]: A large Transformer (360.5M) pretrained with MPM on a large mixed dataset ($\sim 300K$).

Table 2. **Results of Classification on ScanObjectNN PB_50_RS [51] and ModelNet40 [55], Few-Shot Learning on ModelNet40, and Part Segmentation on ShapeNetPart [57].** We report trainable parameters #P (M), inference FLOPs #F (G), overall accuracy (%), class mIoU (%), and instance mIoU (%). We compare methods using $\diamond$ hierarchical architectures, $\circ$ Transformers, and $\star$ Mamba-based models for 3D understanding. $\dagger$: with voting strategy. Note that MoST uses Point Monarch with $b = 16$ by default. All PEFT results are without voting.

Method	Shape Classification				Few-Shot Learning				Part Segmentation	
	#P $\downarrow$	#F $\downarrow$	PB_50_RS $\uparrow$	ModelNet40 $\uparrow$	5w10s $\uparrow$	5w20s $\uparrow$	10w10s $\uparrow$	10w20s $\uparrow$	mIoU _C $\uparrow$	mIoU _I $\uparrow$
<i>Supervised Learning Only (Dense Training)</i>										
$\diamond$ PointNet [35]	3.5	0.5	68.0	89.2	52.0 $\pm$ 3.8	57.8 $\pm$ 4.9	46.6 $\pm$ 4.3	35.2 $\pm$ 4.8	80.4	83.7
$\diamond$ PointNet++ [36]	1.5	1.7	77.9	90.7	38.5 $\pm$ 16.0	42.4 $\pm$ 14.2	23.1 $\pm$ 7.0	18.8 $\pm$ 5.4	81.9	85.1
$\diamond$ DGCNN [54]	1.8	2.4	78.1	92.9	31.6 $\pm$ 2.8	40.8 $\pm$ 4.6	19.9 $\pm$ 2.1	16.9 $\pm$ 1.5	82.3	85.2
$\circ$ Transformer [53]	22.1	4.8	77.2	91.4	87.8 $\pm$ 5.2	93.3 $\pm$ 4.3	84.6 $\pm$ 5.5	89.4 $\pm$ 6.3	83.4	85.1
$\star$ Mamba3D [20]	16.9	3.9	91.8	93.4	92.6 $\pm$ 3.7	96.9 $\pm$ 2.4	88.1 $\pm$ 5.3	93.1 $\pm$ 3.6	83.7	85.7
<i>Supervised Learning Only (Sparse Training)</i>										
$\circ$ MoST (Transformer)	5.6	2.8	87.47	92.83	92.0 $\pm$ 5.8	95.4 $\pm$ 2.9	84.3 $\pm$ 4.5	88.8 $\pm$ 4.3	82.93	85.08
$\star$ MoST (Mamba3D)	5.1	2.6	90.56	92.91	94.4 $\pm$ 2.2	96.0 $\pm$ 2.6	88.3 $\pm$ 5.2	89.4 $\pm$ 4.7	83.11	85.21
<i>with Pre-Training (Full Fine-Tuning)</i>										
$\circ$ Point-BERT [58]	22.1	4.8	83.07	93.2	94.6 $\pm$ 3.1	96.3 $\pm$ 2.7	91.0 $\pm$ 5.4	92.7 $\pm$ 5.1	84.1	85.6
$\circ$ Point-MAE [34]	22.1	4.8	85.18	93.8	96.3 $\pm$ 2.5	97.8 $\pm$ 1.8	92.6 $\pm$ 4.1	95.0 $\pm$ 3.0	84.2	86.1
$\diamond$ Point-M2AE [61]	15.3	3.6	86.43	94.0	96.8 $\pm$ 1.8	98.3 $\pm$ 1.4	92.3 $\pm$ 4.5	95.0 $\pm$ 3.0	84.9	86.7
$\diamond$ I2P-MAE [62]	15.3	3.6	90.11	93.7	97.0 $\pm$ 1.8	98.3 $\pm$ 1.3	92.6 $\pm$ 5.0	95.5 $\pm$ 3.0	85.2	86.8
$\circ$ ACT [38]	22.1	4.8	88.21	93.7	96.8 $\pm$ 2.3	98.0 $\pm$ 1.4	93.3 $\pm$ 4.0	95.6 $\pm$ 2.8	84.7	86.1
$\circ$ ReCon [38]	43.6	5.3	90.63	94.1	97.3 $\pm$ 1.9	98.9 $\pm$ 1.2	93.3 $\pm$ 3.9	95.8 $\pm$ 3.0	84.8	86.6
$\circ$ PointGPT [4] †	360.5	67.7	93.4	94.7	98.0 $\pm$ 1.9	99.0 $\pm$ 1.0	94.1 $\pm$ 3.9	96.1 $\pm$ 2.8	84.7	86.4
$\star$ PointMamba [33]	12.3	3.6	89.31	93.6	95.0 $\pm$ 2.3	97.3 $\pm$ 1.8	91.4 $\pm$ 3.3	92.8 $\pm$ 4.0	84.4	86.0
$\star$ Mamba3D [20]	16.9	3.9	92.05	94.7	96.4 $\pm$ 2.2	98.2 $\pm$ 1.2	92.4 $\pm$ 4.1	95.2 $\pm$ 2.9	84.1	85.7
$\circ$ ReCon++ [39] †	657.2	-	95.25	94.8	98.0 $\pm$ 2.3	99.5 $\pm$ 0.8	94.5 $\pm$ 4.1	96.5 $\pm$ 3.0	-	-
<i>with Pre-Training (Parameter-Efficient Fine-Tuning)</i>										
$\circ$ ReCon [38]	22.1	4.8	90.63	94.1	97.3 $\pm$ 1.9	98.9 $\pm$ 1.2	93.3 $\pm$ 3.9	95.8 $\pm$ 3.0	84.52	86.1
+ IDPT [60]	1.7	7.2	88.13	93.6	96.9 $\pm$ 2.4	98.3 $\pm$ 0.7	92.8 $\pm$ 4.0	95.5 $\pm$ 3.2	83.66	85.7
+ DAPT [66]	1.1	5.0	89.38	93.5	95.6 $\pm$ 2.8	97.7 $\pm$ 1.6	91.9 $\pm$ 4.1	94.6 $\pm$ 3.5	83.87	85.7
+ PPT [63]	1.1	11.3	89.52	93.8	97.0 $\pm$ 2.7	98.7 $\pm$ 1.6	92.2 $\pm$ 5.0	95.6 $\pm$ 2.9	84.23	85.6
+ PointGST [32]	0.6	4.8	89.49	93.6	96.2 $\pm$ 2.7	98.2 $\pm$ 1.1	92.1 $\pm$ 4.4	95.6 $\pm$ 2.8	83.98	85.8
+ MoST	1.3	4.8	92.85	94.7	97.1 $\pm$ 2.0	98.9 $\pm$ 0.9	93.1 $\pm$ 3.8	95.6 $\pm$ 2.8	84.42	86.0
$\star$ Mamba3D [20]	16.9	3.9	92.05	94.7	96.4 $\pm$ 2.2	98.2 $\pm$ 1.2	92.4 $\pm$ 4.1	95.2 $\pm$ 2.9	84.13	85.7
+ IDPT [60]	1.7	6.2	87.34	93.2	95.9 $\pm$ 2.3	97.6 $\pm$ 1.5	91.8 $\pm$ 4.6	94.9 $\pm$ 3.2	82.98	85.3
+ DAPT [66]	1.1	4.0	88.55	92.9	96.0 $\pm$ 2.0	97.7 $\pm$ 1.7	92.1 $\pm$ 4.5	94.8 $\pm$ 3.7	83.49	85.5
+ PPT [63]	1.1	8.5	87.61	92.87	96.2 $\pm$ 2.7	97.7 $\pm$ 1.8	91.7 $\pm$ 4.7	94.7 $\pm$ 3.6	83.19	85.4
+ PointGST [32]	0.6	3.9	89.97	93.7	96.5 $\pm$ 2.1	97.5 $\pm$ 1.6	92.4 $\pm$ 4.7	94.9 $\pm$ 3.1	83.39	85.5
+ MoST	1.2	3.9	92.85	94.7	96.8 $\pm$ 2.4	98.3 $\pm$ 1.8	93.0 $\pm$ 4.5	95.4 $\pm$ 3.0	83.74	85.7

4.2. Implementation Details

We follow the full fine-tuning experimental setup but freeze the pretrained backbone, making only Point Monarch and the task header trainable. Unless otherwise specified, we use Point Monarch with a block count $b = 16$ and K -Rectify with $K = 4$. To optimize spectral norm at initialization and stabilize training while scaling b , we initialize $\mathbf{L} = 0$ and $\mathbf{R} = \mathcal{N}(0, \sigma^2)$, where $\sigma = \sqrt{\min(d_{\text{in}}, d_{\text{out}})/(d_{\text{in}})^2}$, based on scaling laws for structured matrices [41]. To reduce hyperparameter tuning, we set λ in K -Rectify as learnable. We reparameterize the linear projections in Transformer’s Attention [53], Mamba’s SSM [16], and the FFN [53] across all models. All experiments are conducted on a single RTX 4090 GPU. See Appendix for more details.

4.3. Comparison on Downstream Tasks

3D Real-World / Synthetic Object Recognition. We evaluate shape classification on the challenging PB_50_RS variant of the real-world ScanObjectNN [51] and the synthetic ModelNet40 [55] datasets. As shown in Tab. 1, we compare two types of PEFT methods: additive (e.g., VPT [24], IDPT [60], DAPT [66]) and selective/reparameterization (e.g., BitFit [59], LoRA [23], and our MoST). MoST consistently delivers the best performance across all models. With only 0.8M (3.6%) trainable parameters ($b = 32$), even PointMAE [34] can surpass full fine-tuning by 6.77% on ScanObjectNN and 0.24% on ModelNet40. Furthermore, with $b = 8$, MoST achieves an average improvement of 3.96% and 1.52% across five models compared to full fine-tuning, while other

Table 3. Comparison results of large-scale indoor scene semantic segmentation on S3DIS Area 5 [1].

Method	Params. (M)	input	mAcc (%) $\uparrow$	mIoU (%) $\uparrow$
<i>Supervised Learning Only</i>				
$\diamond$ PointNet [†] [35]	3.6	xyz+rgb	49.0	41.1
$\diamond$ PointCNN [†] [35]	3.6	xyz+rgb	63.9	57.3
$\diamond$ PointNet++ [†] [36]	1.0	xyz+rgb	67.1	53.5
$\diamond$ PCT [†] [17]	2.9	xyz+rgb	67.7	61.3
$\circ$ Transformer [53]	27.0	xyz	68.6	60.0
$\star$ Mamba3D [20]	21.9	xyz	67.8	58.0
<i>Supervised Learning Only (Sparse Training)</i>				
$\circ$ MoST (Transformer)	5.3	xyz	64.7	51.4
$\star$ MoST (Mamba3D)	5.9	xyz	67.6	57.1
<i>with Pretraining (Full Finetuning)</i>				
$\circ$ Point-BERT [58]	27.0	xyz	69.7	60.5
$\circ$ Point-MAE [34]	27.0	xyz	69.9	60.8
$\circ$ ReCon [38]	27.0	xyz	69.7	60.8
$\star$ Mamba3D [20]	21.9	xyz	68.7	60.7
<i>with Pretraining (Efficient Finetuning)</i>				
$\circ$ ReCon [38]	27.0	xyz	69.7	60.8
+ Linear Probing	5.2	xyz	64.3	51.2
+ IDPT [60]	5.6	xyz	62.9	50.5
+ DAPT [66]	5.6	xyz	66.3	56.3
+ PPT [63]	5.6	xyz	65.6	54.8
+ PointGST [32]	5.6	xyz	67.8	57.9
+ MoST	5.8	xyz	68.8	58.9

PEFT methods like PPT [63] and LoRA [23] fall short. In particular, MoST allows PointGPT [4] to reach 97.50% accuracy on ScanObjectNN and 96.23% on ModelNet40, setting new state-of-the-art results. These results underscore the effectiveness and strong generalizability of MoST.

Few-Shot Learning. We follow previous work [43] to conduct few-shot learning experiments on ModelNet40 [55], using an n -way, m -shot setting with $n \in \{5, 10\}$ and $m \in \{10, 20\}$. As shown in Tab. 2, MoST fine-tuned on Mamba3D [20] and ReCon [38] outperforms existing 3D PEFT methods, demonstrating its effectiveness. Furthermore, when using sparse training—replacing dense layers with Point Monarch and training from scratch—MoST boosts the performance of both Transformer [53] and Mamba3D. For Transformer, MoST achieves an average improvement of 1.35% acc. while reducing the parameter count to 1/5.

3D Part Segmentation. We conduct part segmentation experiments on ShapeNetPart [57], reporting both average instance IoU (mIoU_I) and average category IoU (mIoU_C), as shown in Tab. 2. MoST fine-tuned on Mamba3D [20] and ReCon [38] surpasses existing 3D PEFT methods and achieves results comparable to full fine-tuning. Even with sparse training, Mamba3D and Transformer [53] achieve mIoU_C of 82.93% and 83.11%, respectively. This fine-grained segmentation experiment further highlights MoST’s ability to capture fine-grained local features, enhancing its capability to process point clouds.

3D Scene Semantic Segmentation. To further verify

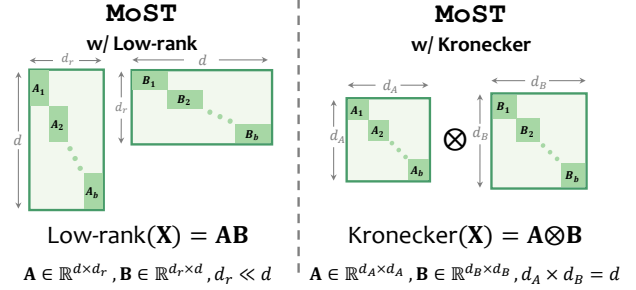


Figure 5. Variants with Low-rank or Kronecker decomposition.

Table 4. Results of MoST variants in different combinations from Fig. 5 with PointGPT [4]. [†]: joint Low-rank decomposition.

Method	Matrix L	Matrix R	Params. (M)	PB_50_RS (%)
I	Low-rank	Low-rank	2.1 (0.6%)	97.02 (-0.17)
II	Kronecker	Kronecker	0.8 (0.2%)	96.08 (-1.11)
III	Low-rank [†]	Low-rank [†]	1.6 (0.4%)	96.88 (-0.31)
IV	L	Low-rank	2.3 (0.6%)	97.09 (-0.10)
V	L	Kronecker	1.7 (0.5%)	97.09 (-0.10)
VI	Low-rank	Kronecker	1.4 (0.4%)	96.18 (-1.01)
MoST _{b=32}	L	R	2.5 (0.7%)	97.19 (+0.00)

MoST’s effectiveness on large-scale scene datasets, we conduct semantic segmentation experiments on the S3DIS Area 5 [1], with results shown in Tab. 3. In this scene segmentation task, most fine-tuning parameters are concentrated in the segment header, resulting in similar parameter counts across methods. Nevertheless, MoST achieves 68.8% mAcc and 58.9% mIoU, outperforming other 3D PEFT methods and coming closest to full fine-tuning performance. When using MoST for sparse training, the Transformer [53] requires only 1/4 of the parameters to achieve 64.7% mAcc and 51.4% mIoU, while Mamba3D[20] achieves 67.6% mAcc and 57.1% mIoU. This experiment demonstrates MoST’s generalization and effectiveness on large-scale scenes.

4.4. Variants

We explore MoST variants that combine Low-rank and Kronecker [52] decompositions, as shown in Fig. 5. Both block-diagonal matrix L and R can be decomposed to two sub-matrices, or we can treat L and R as the results of a joint Low-rank decomposition. Including hybrid methods, we verify 6 variants, with results in Tab. 4. The Low-rank and Kronecker decompositions reduce parameters to 2.1M (0.6%) and 0.8M (0.2%) with only a slight performance drop. Combined, they minimize parameters to 1.4M (0.4%) and achieve an accuracy of 96.18%, highlighting the flexibility of MoST in balancing performance and efficiency.

4.5. PEFT vs. 3D PEFT

Tab. 5 compares various PEFT methods using PointMAE [34] on the ScanObjectNN PB_50_RS dataset [51], including those for text, images, and 3D point clouds. Due

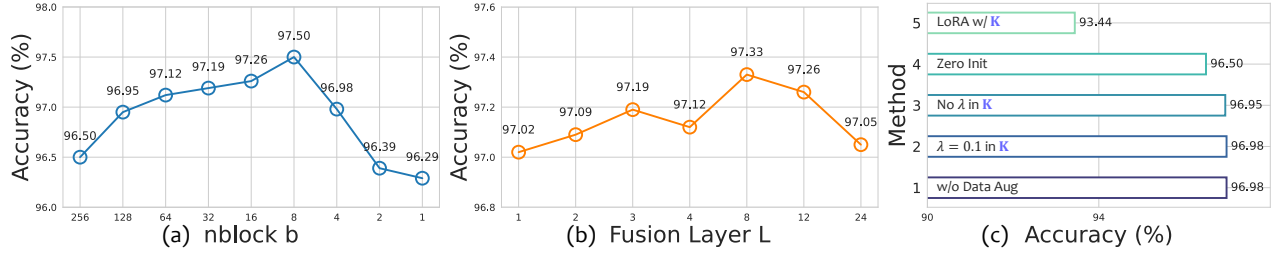


Figure 6. **Ablation of parameters b , L , and others.** We report classification results (%) on ScanObjectNN PB_50_RS using PointGPT [4].

Table 5. Comparisons of PEFT vs. 3D PEFT on the PB_50_RS [51] using Point-MAE [34]. NO: no inference overhead.

Method	Domain	NO	#P (M)	SONN (%)
Point-MAE [34]	-	-	22.1 (100%)	85.18 (+0.00)
+ Linear probing	-	-	0.3 (1.4%)	75.99 (-9.19)
+ Adapter [22]	1D	✗	0.9 (4.1%)	83.93 (-1.25)
+ Perfix tuning [28]	1D	✗	0.7 (3.2%)	77.72 (-7.46)
+ BitFit [59]	1D	✓	0.3 (1.4%)	82.62 (-2.56)
+ LoRA [23]	1D	✓	0.9 (4.1%)	82.76 (-2.42)
+ DePT [45]	1D	✗	0.3 (1.4%)	79.70 (-5.48)
+ FourierFT [15]	1D	✓	0.3 (1.4%)	78.57 (-6.61)
+ VPT-Deep [24]	2D	✗	0.4 (1.8%)	81.09 (-4.09)
+ AdaptFormer [5]	2D	✗	0.9 (4.1%)	83.45 (-1.73)
+ SSF [31]	2D	✗	0.4 (1.8%)	82.58 (-2.60)
+ FacT [25]	2D	✓	0.5 (2.3%)	78.76 (-6.42)
+ BI-AdaptFormer [26]	2D	✗	0.4 (2.3%)	83.66 (-1.52)
+ SCT [65]	2D	✗	0.3 (1.4%)	80.40 (-4.78)
+ IDPT [60]	3D	✗	1.7 (7.7%)	88.34 (+3.16)
+ DAPT [66]	3D	✗	1.1 (5.0%)	88.27 (-1.25)
+ PPT [63]	3D	✗	1.1 (5.0%)	89.00 (+3.09)
+ PointGST [32]	3D	✗	0.6 (2.7%)	89.3 (+4.12)
+ MoST _{b=32}	3D	✓	0.8 (3.6%)	91.95 (+6.77)

to the inherent irregularity of point clouds, traditional PEFT methods for text and images perform poorly, with the best Adapter [22] achieving only 83.93% accuracy. In contrast, 3D PEFT methods reach at least 88.27% accuracy, with our MoST achieving 91.95% using only 0.9M fine-tuned parameters, significantly surpassing full fine-tuning.

4.6. Ablation Study

Components. We perform ablation studies on each component using PointGPT [4] on the ScanObjectNN PR_50_RS classification dataset [51], as shown in Tab. 6. Removing K results in a 2.88% accuracy drop, while eliminating Point Monarch entirely causes a 6.8% decrease. The Feature Fusion shows that both 2^{i-1} weighting and MixPool enhance accuracy. Using only Monarch yields 92.68% acc., significantly lower than Point Monarch’s 96.81%. Notably, accuracy drops to 90.90% when removing Monarch, which requires only 0.7M parameters to finetune (0.2%). These results validate the effectiveness of MoST and underscore Point Monarch’s key role in capturing local information.

Parameters. We evaluate the impact of block number b in Point Monarch and layer number L in Feature Fusion

Table 6. Ablation study using PointGPT [4] on ScanObjectNN [51], where “w/o Monarch” denotes fine-tuning only the task header.

Point Monarch			Feature Fusion		#P	PB_50_RS (%)
K	IDW	Monarch	2^{i-1}	MixPool		
✓	-	✓	✓	✓	2.5 (0.7%)	96.77 (-0.42)
✓	✓	-	✓	✓	0.7 (0.2%)	90.90 (-6.29)
-	-	✓	✓	✓	2.5 (0.7%)	94.31 (-2.88)
-	-	-	✓	✓	0.7 (0.2%)	90.39 (-6.80)
✓	✓	✓	-	✓	2.5 (0.7%)	97.05 (-0.14)
✓	✓	✓	✓	-	2.5 (0.7%)	96.91 (-0.28)
✓	✓	✓	-	-	2.5 (0.7%)	96.81 (-0.38)
-	-	✓	-	✓	2.5 (0.7%)	94.03 (-3.16)
-	-	✓	✓	-	2.5 (0.7%)	93.93 (-3.26)
-	-	✓	-	-	2.5 (0.7%)	92.68 (-4.51)
✓	✓	✓	✓	✓	2.5 (0.7%)	97.19 (+0.00)

on model performance, as shown in Fig. 6(a-b). Results indicate that accuracy initially increases with b and peaks at $b = 8$ before declining. At $b = 256$, the model still achieves 96.5% acc. with only 0.9M parameters, demonstrating Point Monarch’s expressiveness. For L in Feature Fusion, $L = 3$ performs well, while peaking at $L = 8$ with 97.33% acc., confirming the effectiveness of multi-layer feature fusion.

Others. Ablations on λ , initialization, LoRA with K, and data augmentation [11] are shown in Fig. 6(c). A learnable λ yields optimal results, while 0-initializing both L and R decreases accuracy. Incorporating K enhances LoRA, though it remains less expressive than Point Monarch. Rotation as data augmentation [11] provides limited improvements.

5. Conclusion

This paper explores reparameterization-based PEFT in 3D representation learning. We introduce MoST, the first reparameterization-based 3D PEFT method tailored for point clouds. At its core, we propose Point Monarch, a novel structured matrix family that uses simple linear transformations to capture local geometric features. MoST reparameterizes dense update weight matrices with local-aware, sparse Point Monarch matrices, achieving state-of-the-art results while maintaining generalizability. Additionally, MoST can combine with matrix decompositions like Low-rank and Kronecker for further parameter reduction. We hope this work will inspire advances in fine-tuning large-scale 3D models.

Acknowledgements.

This work was supported by the China National Natural Science Foundation No. 62202182.

References

- [1] 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 *CVPR*, pages 1534–1543, 2016. 2, 7
- [2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 1
- [3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5
- [4] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *NeurIPS*, 36, 2024. 1, 2, 5, 6, 7, 8
- [5] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, 35:16664–16678, 2022. 8
- [6] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *ICML*, pages 1517–1527. PMLR, 2019. 3, 4
- [7] Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *ICML*, pages 4690–4721. PMLR, 2022. 2, 3, 4
- [8] Christopher De Sa, Albert Cu, Rohan Puttagunta, Christopher Ré, and Atri Rudra. A two-pronged progress in structured dense matrix vector multiplication. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1060–1079. SIAM, 2018. 3, 4
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019. 1, 2
- [10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023. 2
- [11] 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 *ICLR*, 2023. 2, 8
- [12] Ben Fei, Liwen Liu, Weidong Yang, Zhijun Li, Wen-Ming Chen, and Lipeng Ma. Parameter efficient point cloud prompt tuning for unified point cloud understanding. *IEEE Transactions on Intelligent Vehicles*, 2024. 3
- [13] Jiajun Fei and Zhidong Deng. Fine-tuning point cloud transformers with dynamic aggregation. In *ICRA*, pages 9455–9462. IEEE, 2024. 3
- [14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018. 4
- [15] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In *ICML*, 2024. 8
- [16] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. 1, 2, 3, 4, 6
- [17] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 7
- [18] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, 43(12):4338–4364, 2020. 1
- [19] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016. 4
- [20] Xu Han, Yuan Tang, Zhaoxuan Wang, and Xianzhi Li. Mamba3d: Enhancing local features for 3d point cloud analysis via state space model. In *ACM MM*, page 4995–5004, 2024. 1, 2, 5, 6, 7
- [21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 2
- [22] 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 *ICML*, pages 2790–2799. PMLR, 2019. 1, 3, 8
- [23] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 1, 2, 3, 5, 6, 7, 8
- [24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 5, 6, 8
- [25] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI*, pages 1060–1068, 2023. 8
- [26] Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *ICCV*, pages 17217–17226, 2023. 8
- [27] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059, 2021. 1, 3

- [28] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597, 2021. 8
- [29] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, pages 828–838, 2018. 1
- [30] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023. 1, 2
- [31] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *NeurIPS*, 35:109–123, 2022. 8
- [32] Dingkan Liang, Tianrui Feng, Xin Zhou, Yumeng Zhang, Zhikang Zou, and Xiang Bai. Parameter-efficient fine-tuning in spectral domain for point cloud learning. *arXiv preprint arXiv:2410.08114*, 2024. 3, 5, 6, 7, 8
- [33] Dingkan 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 *NeurIPS*, 2024. 1, 6
- [34] Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022. 1, 2, 5, 6, 7, 8
- [35] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017. 1, 4, 6, 7
- [36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 1, 4, 6, 7
- [37] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *CVPR*, pages 6134–6144, 2021. 1
- [38] 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 *ICML*, pages 28223–28243. PMLR, 2023. 2, 5, 6, 7
- [39] Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. Shapellm: Universal 3d object understanding for embodied interaction. In *ECCV*, pages 214–238. Springer, 2025. 2, 6
- [40] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NeurIPS*, 2022. 1
- [41] Shikai Qiu, Andres Potapczynski, Marc Anton Finzi, Micah Goldblum, and Andrew Gordon Wilson. Compute better spent: Replacing dense layers with structured matrices. In *ICML*, 2024. 3, 6
- [42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 1, 2, 3
- [43] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. In *NeurIPS*, 2020. 7
- [44] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, pages 770–779, 2019. 1
- [45] Zhengxiang Shi and Aldo Lipani. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. In *ICLR*, 2024. 8
- [46] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. *NeurIPS*, 28, 2015. 3
- [47] Hongyu Sun, Yongcai Wang, Wang Chen, Haoran Deng, and Deying Li. Parameter-efficient prompt learning for 3d point cloud understanding. In *ICRA*, 2024. 3
- [48] Yiwen Tang, Ray Zhang, Zoey Guo, Xianzheng Ma, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Pointpeft: Parameter-efficient fine-tuning for 3d pre-trained models. In *AAAI*, pages 5171–5179, 2024. 2
- [49] Yiwen Tang, Ray Zhang, Jiaming Liu, Zoey Guo, Bin Zhao, Zhigang Wang, Peng Gao, Hongsheng Li, Dong Wang, and Xuelong Li. Any2point: Empowering any-modality large models for efficient 3d understanding. In *ECCV*, pages 456–473. Springer, 2025. 3
- [50] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotequi, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6410–6419. IEEE, 2019. 1
- [51] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *CVPR*, pages 1588–1597, 2019. 2, 3, 5, 6, 7, 8
- [52] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2): 85–100, 2000. 2, 5, 7
- [53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 1, 2, 3, 4, 6, 7
- [54] 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 TOG*, 38(5): 146:1–146:12, 2019. 1, 3, 6
- [55] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 2, 5, 6, 7
- [56] Aoran Xiao, Jiaxing Huang, Dayan Guan, Xiaoqin Zhang, Shijian Lu, and Ling Shao. Unsupervised point cloud representation learning with deep neural networks: A survey. *IEEE TPAMI*, 45(9):11321–11339, 2023. 1, 2
- [57] 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 TOG*, 35(6):1–12, 2016. 2, 6, 7
- [58] 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 *CVPR*, 2022. 1, 2, 6, 7

- [59] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9, 2022. [1](#), [5](#), [6](#), [8](#)
- [60] 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 *ICCV*, pages 14161–14170, 2023. [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)
- [61] 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 *NeurIPS*, 2022. [1](#), [6](#)
- [62] 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 *CVPR*, pages 21769–21780, 2023. [2](#), [5](#), [6](#)
- [63] Shaochen Zhang, Zekun Qi, Runpei Dong, Xiuxiu Bai, and Xing Wei. Positional prompt tuning for efficient 3d representation learning. *arXiv preprint arXiv:2408.11567*, 2024. [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)
- [64] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018. [3](#)
- [65] Henry Hengyuan Zhao, Pichao Wang, Yuyang Zhao, Hao Luo, Fan Wang, and Mike Zheng Shou. Sct: A simple baseline for parameter-efficient fine-tuning via salient channels. *IJCV*, 132(3):731–749, 2024. [8](#)
- [66] Xin Zhou, Ding kang 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 *CVPR*, pages 14707–14717, 2024. [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)
- [67] Haoyi Zhu, Yating Wang, Di Huang, Weicai Ye, Wanli Ouyang, and Tong He. Point cloud matters: Rethinking the impact of different observation spaces on robot learning. In *NeurIPS*, 2024. [1](#)