FastPoint: Accelerating 3D Point Cloud Model Inference via Sample Point Distance Prediction

Donghyun Lee[1] Dawoon Jeong[1] Jae W. Lee[1] Hongil Yoon[1,2]
[1]Seoul National University [2]Google
{eudh1206, daun20211, jaewlee}@snu.ac.kr, hongilyoon@google.com

A. Supplementary Materials for MDPS

A.1. MDPS Algorithm

In this section, we present detailed algorithm (Algorithm 1) of Minimum Distance Prediction Sampling (MDPS).

1. Minimum Distance Curve Estimation Minimum distance curve estimation starts by performing FPS for 1/10 of the original number of iterations. The key difference between original FPS and this operation (Line 3-11) is that the maximum of minimum distance (i.e., `max`(dists)) must be saved at each iteration. These values serve as input to the minimum distance curve estimator, which predicts the rest of the curve. The computed and estimated values are then concatenated to form the complete curve (Line 13).

2. Distance Curve Segmentation We divide the distance curve into $nseg$ segments and find the points within a specified radius of segment boundaries (i.e., mdc[$n * seg/nseg$]) for each input point. Distance between an input point $P[i]$ and a query point $P[j]$ is calculated (Line 18), and the index of query point is added to the exclusion list if the distance is smaller than the corresponding threshold (Line 19-21). Distance computation between points (Line 18) is performed outside the loop, ensuring that the amount of distance computation remains independent of $nseg$.

3. Sampling with Predicted Distance After initializing the bitmap for all segments to 1 (Line 24), sampling begins. Sampling consists of three main stages: bitmap update, sampling, and sampling availability check. First, we check the entry of exclusion list that corresponds to the previously selected point (Line 30) and update the bitmap with value 1 according to the entry (Line 31-32). Note that we update not only the bitmap of current segment, but also the bitmap of subsequent segments. This ensures that the bitmaps for all subsequent segments remain up to date, facilitating smooth transitions to the next segment during sampling. Next, we sample any point that is available by finding the index of a bitmap entry with a value of 1 (Line 33). If no such entry is found, the process moves to the next segment by incrementing the value seg (Line 34-36). When the final segment is exhausted, we terminate the sampling process and proceed to Early Termination stage (Line 26-28).

4. Early Termination If the sampling stage terminates before acquiring the desired number of points n, we make a transition to Farthest Point Sampling (FPS). For this transition, the FPS distance matrix is initialized with the minimum distances between the input points and the already sampled point set. To optimize this process, instead of computing all-to-all distances, we leverage the exclusion list from the first segment (i.e., excl_list$_1$) to limit the search space when identifying the closest sampled point for each input point (Lines 43–46). Once the distance matrix is updated, the standard FPS algorithm is applied to complete the remaining iterations (Lines 48–53).

A.2. Analysis on Exclusion List Construction

In this section, we analyze the computational complexity of exclusion list construction and compare it with that of the baseline FPS. The exclusion list construction requires all-to-all distance calculations, resulting in a computational complexity of $O(N^2)$, while FPS has a complexity of $O(Nn)$, where N and n denote the number of input points and sampled points, respectively.

Despite the higher theoretical complexity, MDPS is significantly faster than FPS due to two key factors: its high degree of parallelism (as discussed in the paper) and its efficient utilization of GPU Streaming Multiprocessors (SMs).

FPS only utilizes single SM since parallelizing the distance matrix update across SMs requires frequent SM-to-SM communication every iteration, incurring considerable latency overhead. In contrast, exclusion list construction has no such restriction and fully leverages available SMs.

Thus, considering the utilization of SMs, the complexity of MDPS becomes $O(N^2/p)$ (where p represents the number of SMs), while the FPS remains bound by $O(N^2/stride)$ (where n=$N/stride$). This translates to

Algorithm 1 Minimum Distance Prediction Sampling

Input P: input point cloud, n: number of points to sample, N: number of original points
Output sampled_idx: indices of sampled points

// 1. Minimum Distance Curve Estimation
// 1-1. 1/10 FPS iterations
sampled_idx[0] $\leftarrow seed$
dists[i] $\leftarrow \infty$ for $i = 0, ..., N-1$
for $i \leftarrow 1$ to $n/10$ **do**
  for $j \leftarrow 0$ to $N-1$ **do** // Parallelized
    $d_{new} \leftarrow$ `dist`(P[sampled_idx[$i-1$]], $P[j]$)
    if dists[j] $> d_{new}$ **then**
      dists[j] $\leftarrow d_{new}$
  sampled_idx[i] $\leftarrow$ `argmax`(dists)
  mdc[i] $\leftarrow$ `max`(dists)
// 1-2. Estimation
mdc $\leftarrow$ `concat`(mdc[$0 : n/10$], $\mathcal{E}$(mdc[$0 : n/10$]))

// 2. Distance Curve Segmentation
for $i \leftarrow 0$ to $N-1$ **do** // Parallelized
  for $j \leftarrow 0$ to $N-1$ **do** // Parallelized
    $d \leftarrow$ `dist`($P[i]$, $P[j]$)
    for $seg \leftarrow 0$ to $nseg$ **do**
      if $d <$ mdc[$n * seg/nseg$] **then**
        excl_list$_{seg}$[i].`append`(j)

// 3. Sampling with Predicted Distance
bitmap$_{seg}$[i] $\leftarrow 1$ for $i = 0, ..., N-1$, $seg = 1, ..., nseg$
for $i \leftarrow 1$ to $n-1$ **do**
  $seg \leftarrow \max(\text{div}(i, n/nseg), seg)$
  if $seg > nseg$ **then**
    last_idx $\leftarrow i$
    break // Early Termination
  for $l \leftarrow seg$ to $nseg$ **do** // Parallelized
    list $\leftarrow$ excl_list$_l$[sampled_idx[$i-1$]]
    for $j \leftarrow 0$ to `len`(list) **do** // Parallelized
      bitmap$_l$[list[j]] $\leftarrow 0$
  sampled_idx[i] $\leftarrow$ `findAnyOne`(bitmap$_{seg}$)
  if sampled_idx[i] $== -1$ **then** // No point available
    $seg \leftarrow seg + 1$
    $i \leftarrow i - 1$

// 4. Early Termination
// 4-1. Distance Matrix Update
for $i \leftarrow 0$ to $N-1$ **do** // Parallelized
  for $j \leftarrow 0$ to `len`(excl_list$_1$[i]) **do**
    if excl_list$_1$[i][j] in sampled_idx **then**
      $d_{new} \leftarrow$ `dist`($P[i]$, P[excl_list$_1$[i][j]])
      if dists[i] $> d_{new}$ **then**
        dists[i] $\leftarrow d_{new}$
// 4-2. Remainder FPS
for $i \leftarrow$ last_idx to $n-1$ **do**
  for $j \leftarrow 0$ to $N-1$ **do** // Parallelized
    $d_{new} \leftarrow$ `dist`(P[sampled_idx[$i-1$]], $P[j]$)
    if dists[j] $> d_{new}$ **then**
      dists[j] $\leftarrow d_{new}$
  sampled_idx[i] $\leftarrow$ `argmax`(dists)

Method	Dataset	MAPE (%)
MLP	S3DIS	1.0194
	ScanNet	0.7663
	SemanticKITTI	1.9318
Power Function	S3DIS	1.4746
	ScanNet	1.8816
	SemanticKITTI	6.6118

Table 1. Comparison of MAPE values for MLP and Power function based estimators across datasets.

considerable speedup considering the large number of SMs in GPUs (81 in RTX 3090) and typically smaller downsampling $stride$ values in point cloud models ($stride \leq 4$ for all OpenPoints library models).

A.3. Minimum Distance Curve Estimator

To develop a minimum distance curve estimator, we explore various models, including power functions and multi-layer-perceptrons (MLPs).

For the power function-based estimator, the equation $y = \frac{a}{x^n}$ yields the best results. Here, n determines the decreasing speed of the curve while a reflects the density of each input point cloud. The parameter n is determined by fitting the power function to the entire curve obtained from the training split of each dataset. In contrast, a is determined dynamically during inference, using only the first 1/10 segment of the curve.

For the MLP-based estimator, we use a lightweight three-layer MLP (32-128-128-64). The estimator is trained to take the first 1/10 segment of the curve (i.e., represented by 32 values) as input, and estimate the remaining curve (i.e., represented by 64 values). Training is performed using input-output pairs derived from the minimum distance curves of each training dataset. Training is conducted for 20 epochs on both S3DIS and ScanNet, and 30 epochs on SemanticKITTI. We use a batch size of 1 and the SGD optimizer with a learning rate of 0.01 for all datasets.

For evaluation, we use the validation split of each dataset and measure Mean Absolute Percentage Error (MAPE) between the predicted curve and the real curve. Results in Table 1 demonstrate that MLP-based method significantly outperforms power function-based methods, especially on SemanticKITTI dataset. These results highlight the MLP-based method's superior ability to handle the varying point density and distribution of outdoor datasets. Based on these results, we adopt the MLP-based method as our primary approach for minimum distance curve estimation.

A.4. Comparison with L-FPS

The L-FPS [6] algorithm is designed to accelerate the FPS process in the training pipeline of PointNet-based models. L-FPS performs FPS once prior to training to find out the minimum distance between the sampled points. L-FPS then

Model	Dataset	Accuracy (mIoU)		
		Baseline (FPS)	Grid Sampling	Diff.
PV-L	S3DIS	71.33	70.19	-1.14
	ScanNet	70.70	69.84	-0.86
	Semantic KITTI	50.91	51.66	+0.75
PMB-L	S3DIS	69.72	69.16	-0.56
	ScanNet	70.86	70.04	-0.82
	Semantic KITTI	52.19	51.28	-0.91

Table 2. Accuracy comparison of models using FPS and Grid Sampling in both training and inference.

filters out points that are closer than this distance threshold to produce the sampling results for each epoch. While this approach effectively speeds up training, it is not suitable for inference, as the minimum distance value cannot be determined in advance in inference scenarios. Additionally, L-FPS relies on a single threshold (i.e., minimum distance at the final iteration of FPS) during filtering, which results in inferior sampling results compared to FPS (i.e., comparable to 1 segment results in Figure 5).

B. Additional Experiments

B.1. Model Performance Comparison with FPS and Grid Sampling

In this section, we train PointNet++ based models with Grid Sampling and compare the model performance with those trained with FPS (i.e., baseline). While Grid Sampling occasionally achieves better results (e.g., PointVector on SemanticKITTI), FPS generally outperforms Grid Sampling in most cases with a significant performance gain. This consistent trend highlights the robustness of FPS in adapting to various datasets and tasks. Given that MDPS closely matches the performance of FPS, it demonstrates a distinct advantage over Grid Sampling in this context.

B.2. Latency Breakdown of MDPS

Table 3 provides a detailed latency breakdown of four major operations involved in the MDPS: Minimum Distance Curve Estimation, Distance Curve Segmentation, Sampling with Predicted Distance, and Early Termination. For all datasets, Minimum Distance Curve Estimation accounts for the largest portion of the overall latency, ranging from approximately 40% to 50%. This is due to the high latency of initial FPS iterations required for estimation. Distance Curve Segmentation and Sampling with Predicted Distance each contribute a smaller portion of the latency (each 20-30%), highlighting the effectiveness of increased parallelism in distance computation and reduced overhead in sampling. Lastly, Early Termination has the smallest portion, comprising less than 10% of the overall latency. This indicates that significant overestimation rarely occurs, underscoring the high accuracy of our minimum distance curve estimator.

Task	S3DIS	ScanNet	Semantic KITTI
Minimum Distance Curve Estimation	48.52%	52.39%	39.48%
Distance Curve Segmentation	23.23%	24.96%	26.78%
Sampling with Predicted Distance	20.09%	20.34%	27.17%
Early Termination	8.16%	2.31%	6.57%

Table 3. Latency breakdown of MDPS on various datasets.

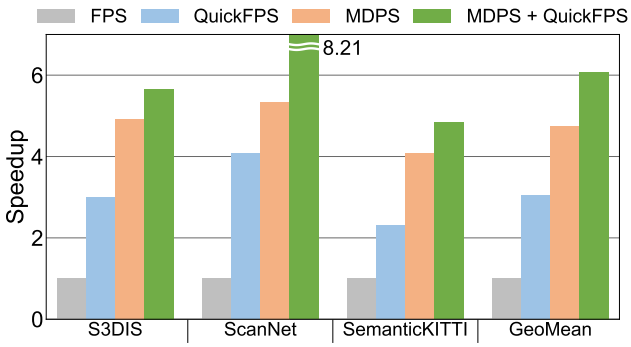


Figure 1. Sampling speedup of MDPS and QuickFPS.

B.3. Speedup of MDPS

Figure 1 demonstrates the speedup specific to the sampling operation. Across the S3DIS [1], ScanNet [3], and SemanticKITTI [2] datasets, FastPoint achieves significant improvements in sampling latency, achieving a geomean speedup of 4.75× compared to FPS. When QuickFPS [4, 5] is integrated with MDPS (i.e., FPS used in minimum distance curve estimation is replaced with QuickFPS), MDPS achieves geomean speedup of 6.08× compared to baseline FPS and 2.00× compared to standalone QuickFPS. The high portion of minimum distance curve estimation in total MDPS latency (Table 3) explains the substantial improvement in sampling speed enabled by the integration of QuickFPS.

B.4. Speedup of Redundancy Free Neighbor Search

In this section, we report the speedup specific to the neighbor search operations (i.e., Ball Query, k-NN). Figure 2 highlights the significant latency reduction for neighbor search operations achieved through Redundancy Free Neighbor Search. Thanks to the reuse of exclusion list

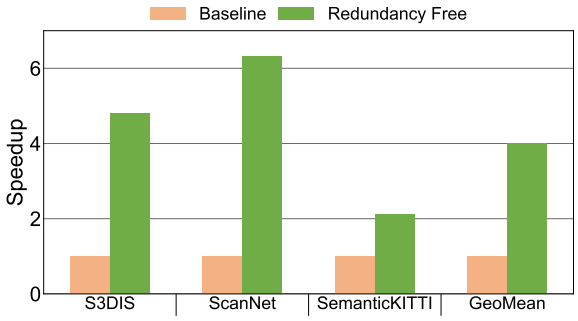


Figure 2. Speedup of Redundancy Free Neighbor Search.

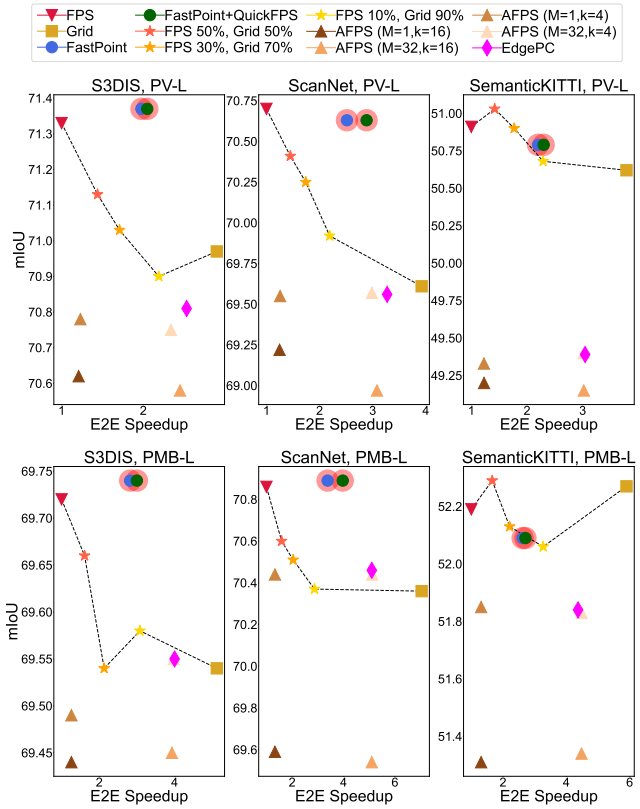


Figure 3. Speedup-mIoU curve of various sampling methods. PV and PMB stand for PointVector and PointMetaBase.

in Ball Query and search space reduction in k-NN, we achieve an impressive geomean speedup of 3.99× across the S3DIS, ScanNet, and SemanticKITTI datasets. These results confirm that FastPoint is not only effective in achieving speedup for sampling but also excels in accelerating neighbor search operations.

B.5. Latency-Accuracy Comparison with Other Sampling Methods

In this section, we compare latency and accuracy of FastPoint with other sampling methods. The speedup-mIoU plot (Figure 3) demonstrates that FastPoint lies above the Pareto front, while both Adjustable FPS [7] and EdgePC [9] exhibit suboptimal performance. Although Grid sampling achieves higher speedup than FastPoint, it suffers notable loss in mIoU compared to FastPoint. For a more comprehensive comparison, we also evaluate a hybrid approach combining two existing state-of-the-art sampling algorithms: FPS and Grid sampling. To the best of our knowledge, FastPoint is the fastest sampling method that maintains both accuracy and sampling quality on par with FPS.

	p=0.2	p=0.1	p=0.05
Est. Error (%)	0.72	0.77	1.46
Smpl. Q. (%)	99.3	99.35	99.01
mIoU (%)	70.95	70.89	70.83
E2E Speedup	2.70×	3.38×	3.82×

Table 4. Ablation study for initial FPS iterations. PointMetaBase-L model and ScanNet dataset is used.

	S3DIS		ScanNet		SemanticKITTI	
	no aug	aug	no aug	aug	no aug	aug
Est. Error (%)	1.02	0.98	0.77	0.93	1.93	2.05
Smpl. Q. (%)	99.45	99.40	99.35	99.10	98.36	98.30
mIoU diff (%)	+0.02	-0.03	+0.03	0	-0.1	-0.11

Table 5. Robustness on augmentation. PointMetaBase-L is used when measuring mIoU.

B.6. Ablation Study for Initial FPS Iterations

We perform an ablation study on the impact of the number of initial FPS iterations used for estimation. Table 4 demonstrates estimator error, sampling quality, model accuracy, and end-to-end speedup for different values of p, where p represents the ratio of the initial FPS iterations to the total iterations. As p decreases, end-to-end speedup improves due to reduced estimation time, while sampling quality and model accuracy decline due to the increased estimator error. To balance error and efficiency, we select $p = 0.1$.

B.7. Robustness of FastPoint

To evaluate the robustness of FastPoint against data augmentations, we apply the same training augmentations (i.e., jitter, rotation, scaling, and point dropping) to the validation set. As shown in Table 5, these augmentations have minimal impact on estimator error, sampling quality, and model accuracy. This is because the minimum distance curve's smoothness is preserved despite the augmentations, which is the key of predictability. Considering that data augmentations used during training typically reflect the real-world variations, the experimental results substantiate the claim that FastPoint is robust in practical scenarios.

	ScanNet Estimator	S3DIS Estimator	SemanticKITTI Estimator
Est. Error (%)	0.77	1.42	2.24
Smpl. Q. (%)	99.35	99.28	98.89
mIoU (%)	70.86	70.92	70.93

Table 6. Cross-dataset applicability of FastPoint. Experiments are performed on PointMetaBase-L model, ScanNet dataset.

	FPS	Grid	FastPoint	FastPoint+QuickFPS
mIoU (%)	70.85	70.34	70.74	70.74
E2E speedup	1×	2.97×	2.16×	2.27×

Table 7. Speedup and mIoU on Point Transformer, S3DIS dataset.

B.8. Cross-Dataset Applicability of FastPoint

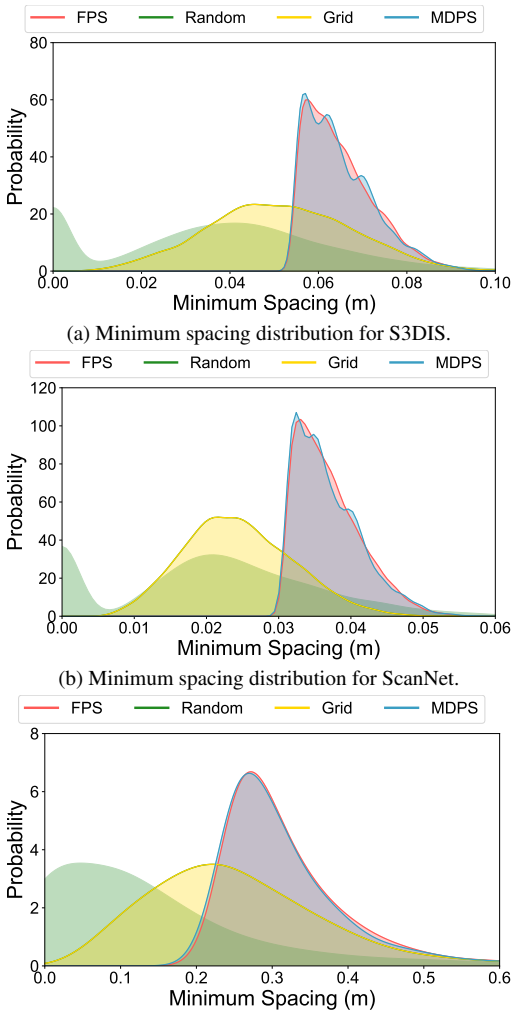
To explore the cross-dataset applicability of estimator, we apply estimator trained on S3DIS and SemanticKITTI dataset to ScanNet dataset. The estimator trained on S3DIS performed well on ScanNet, while the SemanticKITTI estimator had a relatively higher error and lower sampling quality, indicating better compatibility within similar indoor datasets but challenges in cross-environment generalization. Still, the sampling quality loss is minimal ($< 2\%$) in all cases, demonstrating no impact on model accuracy.

B.9. Visualizing Minimum Spacing Distribution

Figure 4 illustrates the distribution of minimum spacing between sampled points using FPS, Random Sampling, Grid Sampling, and MDPS. MDPS demonstrates a distribution highly similar to that of FPS, confirming its effectiveness in resembling the high sampling quality of FPS. In contrast, Grid Sampling and Random Sampling exhibit distributions that deviate significantly from FPS, indicating lower sampling quality.

B.10. Scalability to Non-PointNet++-Based Models

FastPoint is a *model-agnostic technique* accelerating FPS and neighbor search. It is applicable to any point cloud models that use these operations. We applied FastPoint to Point Transformer model adopting FPS [11], achieving 2.16× end-to-end speedup without sacrificing accuracy as shown in Table 7. Furthermore, several recent Mamba-based models with FPS [8, 10] achieve state-of-the-art performance on various datasets, substantiating that FastPoint has the potential to accelerate a wide range of emerging point cloud models.



(a) Minimum spacing distribution for S3DIS.

(b) Minimum spacing distribution for ScanNet.

(c) Minimum spacing distribution for SemanticKITTI.

Figure 4. Minimum spacing distribution of each sampling method for S3DIS, ScanNet, and SemanticKITTI.

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*, 2016. 3

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 3

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3

[4] Meng Han, Liang Wang, Limin Xiao, Hao Zhang, Chenhao Zhang, Xiangrong Xu, and Jianfeng Zhu. Quickfps. http://github.com/hanm2019/bucket-based_farthest-point-sampling_GPU. 3

[5] Meng Han, Liang Wang, Limin Xiao, Hao Zhang, Chenhao

Zhang, Xiangrong Xu, and Jianfeng Zhu. Quickfps: Architecture and algorithm co-design for farthest point sampling in large-scale point clouds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023. 3

[6] Donghyun Lee, Yejin Lee, Jae W. Lee, and Hongil Yoon. Frugal 3d point cloud model training via progressive near point filtering and fused aggregation. In *ECCV*, 2024. 2

[7] Jingtao Li, Jian Zhou, Yan Xiong, Xing Chen, and Chaitali Chakrabarti. An adjustable farthest point sampling method for approximately-sorted point cloud data. In *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, 2022. 4

[8] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. In *NeurIPS*, 2024. 5

[9] Ziyu Ying, Sandeepa Bhuyan, Yan Kang, Yingtian Zhang, Mahmut T. Kandemir, and Chita R. Das. Edgepc: Efficient deep learning analytics for point clouds on edge devices. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*, 2023. 4

[10] Tao Zhang, Haobo Yuan, Lu Qi, Jiangning Zhang, Qianyu Zhou, Shunping Ji, Shuicheng Yan, and Xiangtai Li. Point cloud mamba: Point cloud learning via state space model. In *AAAI*, 2025. 5

[11] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 5