VoxelMamba : Voxelization in Mamba-State Space Model for Point Cloud Processing

***Abstract*—As the application domain of Machine Perception expands, so does the demand for efficient and accurate architectures for point cloud processing, especially for real-time applications such as Autonomous Driving. Most of the existing methods lack the ability to capture global features of unstructured data like point clouds effectively. The most accurate state-of-the-art methods based on transformers have quadratic time complexity, hence do not scale very well. The recent introduction of SSM-based Mamba architecture presents a flexible module with significant potential to enhance existing deep learning architectures. It offers accuracy comparable to transformers while maintaining linear time complexity. However, its utilization in 3D deep learning, particularly in point cloud processing, remains largely unexplored. In this paper, we introduce a novel architecture that leverages the Mamba-SSM module for general point cloud processing tasks, including classification and segmentation. We amalgamate the Transformation-Invariance module with voxelization methods for linear-time point cloud embedding. This embedded representation is then fed into the Mamba-SSM module and MLP/Convolutional feature upscaling module to construct the encoder. The architecture is designed with a flexible output head to accommodate various tasks. Experimental results demonstrate the efficacy of the proposed architecture in point cloud processing tasks. Its accuracy reaches up to 93.4% in point cloud classification on the ModelNet40 dataset. Additionally, it attains mean Accuracy (mAcc) scores of 67.94 and mean Intersection over Union (mIoU) scores of 61.75 for segmentation tasks on the S3DIS Dataset, all while maintaining linear time complexity.**

***Impact Statement*—Point clouds serve as a foundational means of representing 3D environments. However, their voluminous data and additional depth dimension often impede real-time processing, rendering many existing techniques inadequate. The proposed methodology seeks to expedite processing while maintaining high accuracy, achieving approximately 5% higher average accuracy in point cloud processing compared to the state-of-the-art methods. Furthermore, it attains a preprocessing pace that surpasses recent methodologies by 20-30%, demonstrating an equilibrium between accuracy and efficiency. With continued refinement, the proposed methodology holds promise for reducing runtime from approximately 200 milliseconds to 100-70 milliseconds on real-world data when executed on high-end devices. Such enhancements could establish this approach as the preeminent method for point cloud processing, offering a viable runtime while significantly broadening the applicability of 3D information in conventional image processing domains such as object and anomaly detection, terrain analysis, construction monitoring, and the creation of 3D structures using 2D images.**

***Index Terms*—State Space Model, Mamba, Voxel, Point Cloud, Grid, Invariance, Classification, Segmentation**

I. Introduction

A. Problem Statement

WITH the expanding applications in domains such as Autonomous Driving, Robotic Vision, and Augmented/Virtual reality, the demand for deep learning architectures for 3D Computer Vision continues to surge. Point clouds, being the simplest representation of 3D structures in a numerical format, have led to the development of numerous deep learning architectures for point cloud processing in recent times. These architectures can be broadly categorized into three main types: CNN-based approaches like PointConv [1], raw point cloud-based models such as PointNet [2], and transformer based architectures like PCT [3].

Notably, point clouds, being unordered and unstructured, can be more effectively processed by transformer based architectures capable of handling sequences of indefinite lengths, and having a global attention mechanism. However, the inherent quadratic complexity of transformers renders them practically infeasible for application on raw point cloud sequences, which generally exceed lengths of 100,000 points in real-world scenarios.

The recent introduction of State-Space Models for vision tasks has shown promising potential as a replacement for transformers, offering the ability to scale linearly for large sequences while capturing long-range dependencies and benefiting from parallel processing. The Mamba-SSM [4]-based Vim [5] and VSS blocks [6] have demonstrated a global transformer-like receptive field while maintaining linear time complexity. However, only a handful of applications of Mamba-SSM can be found in the field of 3D Computer Vision, primarily differing in the way point clouds are encoded for processing. Utilizing Mamba-SSM with voxelization techniques for point cloud embedding remains an unexplored area, forming the basis of the research work presented in this paper. We propose an architecture that employs voxelization techniques to embed point clouds for processing via the Mamba-SSM module, incorporating the Transformation-Invariance block from the PointNet [2] architecture and MLPs for feature scaling to accelerate point cloud processing. With a modular head for downstream tasks, we conduct extensive experiments using the proposed architecture for segmentation and classification tasks on publicly available datasets and a custom 108-point cloud testing dataset. Furthermore, we evaluate our model's correctness using various metrics and provide claimed efficiency plots to depict its scalability.

B. Contributions and Organization of the Paper

We propose a novel Mamba-SSM based architecture that uses voxelization to convert point cloud into sequences for linear processing. The contributions can be summarised as follows :

1) A novel architecture that incorporates the Transformation-Invariance module from PointNet followed by a voxelization technique to select only non-empty voxels, creating a sequential representation

of the point cloud. The point cloud is passed through a backbone that uses Mamba-SSM for sequence processing, followed by an MLP for feature upscaling.
2) We use two different output heads attached to the backbone and conduct extensive experiments on our proposed architecture for classification and segmentation tasks, on public datasets as well as a custom 100 point cloud dataset.
3) We demonstrate the effectiveness of our model in different real-world scenarios, comparing it with state-of-the-art models and analyzing the efficiency of our model.

Next section describes the related works in the field followed by the section explaining methodology adopted for point cloud processing and the reasoning for the same. Section IV presents the experimental analysis along with the obtained results from our approach, followed by Section V which demonstrates the impact of individual architectural designs on the model's statistics. Finally, Section VI concludes the research and outlines potential future work.

II. Background

Deep Learning methods in 3D Computer Vision have gradually evolved from light, fast, and less accurate CNN-based methods to heavy, slow, and more accurate transformer-based methods. Throughout this transition, all architectures propose a shared trait of being able to process varying size inputs, since point clouds are inherently unstructured and unordered.

1) CNN-based Methods: One way to process such unstructured data is by applying a series of convolution operations. The success of 2D CNNs in feature extraction led to their usage in 3D tasks as well. Earliest methods utilize the multi-view projections of 3D data to convert it into a 2D representable format, which could be modeled using standard 2D CNNs [7]–[9]. This projection inevitably caused the loss of 3D spatial information, such as depth, necessitating the development of methods that can directly operate on 3D data. In this context, 3D CNNs could be implemented by aggregating the 2D information with depth, termed voxelization methods [10], [11], which made the 3D data directly operable by deep learning architectures.

2) Point-based Methods: The overhead of preprocessing point clouds, such as projection or Voxelization, could be surmounted by using raw point clouds as input. Specifically, PointNet [2] and PointNet++ [12] utilized Permutation-Invariant and Transformation-Invariant modules to overcome the ubiquitous unstructuredness of point clouds. Since the information of a single point in a point cloud is useless, these methods use the KNN algorithm to extract the relative spatial information of a point with respect to its neighbors. However, using KNN results in a greatly increased time complexity for computing pairwise distances between points.

3) Graph-based Methods: Other approaches for processing point clouds are based on converting the points into a graph structure, such as DGCNN [13], which performs convolutional operations on large-scale labeled graphs using the KNN algorithm, or SPG [14], which operates on a superpoint graph. However, these approaches incur heavy overhead due to the requirement of preprocessing point clouds into superpoint graphs and heavy dependence on geometrically homogeneous partitioning, which sometimes may even lead to incorrect segmentation.

4) Transformer-based Methods: Transformer-based architectures, such as ViT [15], DeiT [16], and Swin Transformer [17], are the most popular architectures currently in the vision field, adept at modeling large-distance global features using a quadratic complexity attention mechanism. This inherent quadratic complexity is unsuited for point cloud applications due to their inability to scale to large sequence lengths. Current methods using transformers employ different approaches to reduce this sequence length, such as grouping points in spatial proximity using KNN, as in PCT [3], using voxelization, as in VoTR [18] and VoxSet [19], or using other ordering methods, such as Z-order/Hilbert [20].

5) State Space Models: State Space Models [21] are potential alternatives to transformers for long-range dependency learning with linear time complexity. To solve the problem of high memory usage by raw SSMs, S4 [22] was introduced, which normalizes parameters into a diagonal structure. Combining with a Selective-Scan mechanism, a general data-dependent SSM backbone architecture, Mamba [4], was introduced for language tasks and was soon followed by its introduction in vision tasks. By embedding Mamba into a transformer-like architecture, VMamba [6] and Vision Mamba [5] were introduced for image processing tasks.

Only a handful of models have been proposed for point cloud processing using the Mamba-SSM module, the primary difference being in the way the points are embedded into the Mamba module. Point Mamba [23], [24] uses methods such as KNN and Octree-based ordering to group points into patches that can be treated as sequences. However, standard voxelization methods for embedding points are still unexplored in the context of Mamba for point cloud processing.

III. Methodology

A. Overview

In this section we describe the methodology adopted for processing the point clouds along with the justifications for each of the architectural module utilized. The initial embedding process is shown in the figure 1, which is then continued by the main network represented in figure 2. Finally a flexible output head is used to complete the entire network of the model.

B. Transformation-Invariance

To make transformations such as scaling, rotation and translation indifferent within any architecture, PointNet [2] first used the Transformation-Invariance module originally proposed for the Spatial Transformer [25]. Due to its flexibility, this module can be integrated into any existing framework or used as a pre-processing step to significantly reduce training time and required dataset size, because of the same input and output dimensions of the module. By making the network transformation invariant, more of the network can focus on feature learning rather than alignment. While some studies

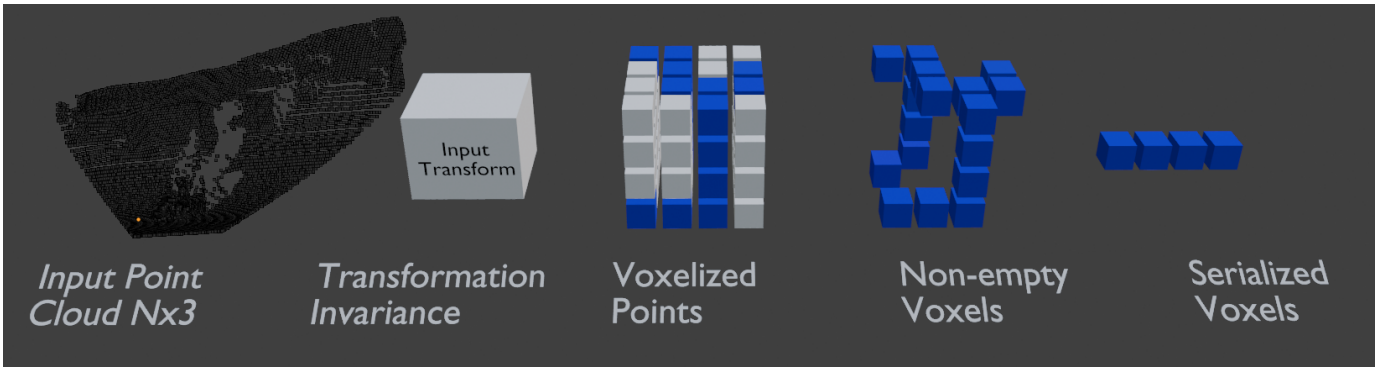


Fig. 1: Embedding Input Point Cloud

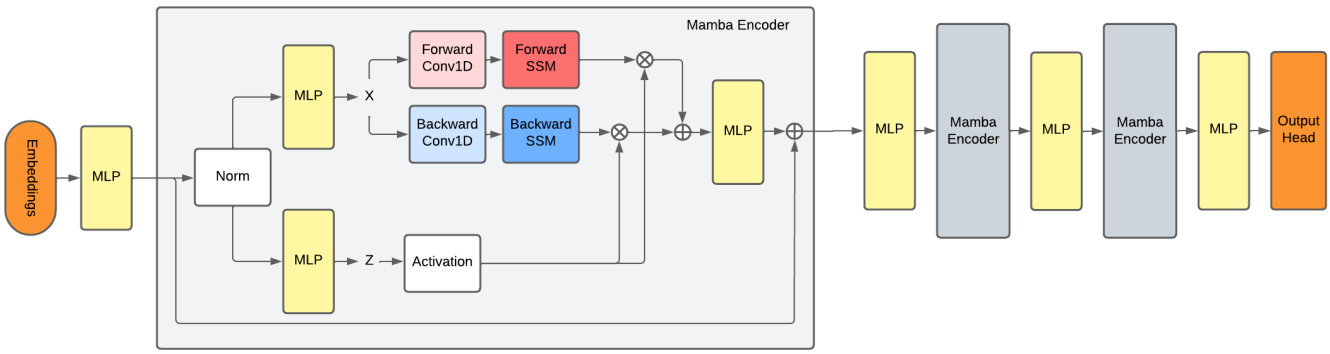


Fig. 2: Main Network Architecture of the proposed model

have shown that Permutation-Invariance is not required for point cloud analysis [26], there is still debate about the importance of Transformation-Invariance.

By including the following loss function component as the main network's loss, the Transformation-Invariance module can be trained alongside the main network:

$$L_{\text{ortho}} = \left|I - A^{\top}A\right|_F^2 \tag{1}$$

$$L_{\text{trans}} = |X - AX|_F^2 \tag{2}$$

$$L_{\text{T-Net}} = L_{\text{ortho}} + L_{\text{trans}} \tag{3}$$

where I is the Identity Matrix, A is the 3x3 Transformation Matrix learned by the T-Net, X is the input point cloud and $|\cdot|_F$ denotes the Frobenius Norm. The network can also be trained independently, or pretrained T-Net models can utilized as a preprocessing step for removing transformations from the point cloud before voxelizing it.

C. Voxelization

The global structure of a point cloud cannot be inferred from a single point. Point clouds can only yield meaningful information when their spatial relationship and that of their neighboring points are taken into account. However, because point clouds typically contain more than 100,000 points, it is practically impossible to apply attention mechanisms to each individual point. One of the most widely used techniques for grouping point clouds is KNN, which is followed by global attention mechanisms on these groups taking into account their spatial structure, as in [24], which sorts these KNN groups on individual axes. But as [20] points out, in such architectures, KNN uses up to 28 % of the forward time, defeating the whole point of Mamba for linear time sequence modeling. Voxelization can be completed in linear time, but improper implementation can result in significant overhead. We convert points to voxels using the following technique:

$$x_{\text{min}} = \min\left(x_i \forall i \in [1, N]\right) \tag{4}$$

$$x_{\text{max}} = \max\left(x_i \forall i \in [1, N]\right) \tag{5}$$

where x_{min} and x_{max} are the mininmum and maximum x-coordinate from all the points in the point cloud. Assuming an arbitrary voxel grid of dimensions $[H, W, D]$ and any point in the point cloud $[x_i, y_i, z_i]$, the voxelized co-ordinates of the point can be given as :

$$x_i = \frac{x_i - x_{\text{min}}}{(x_{\text{max}} - x_{\text{min}})/H} \tag{6}$$

This corresponds to scaling the range $x_{\text{max}} - x_{\text{min}}$ to the range H, where H is the dimension of the grid along the x axis. To obtain the exact coordinates of a point in the voxel grid, a similar process can be repeated for the coordinates y and z. This process takes $O(3N)$ time, where N is the total

number of points in the point cloud. Additionally, a reverse hashmap is used to count the total number of points in each voxel. Later, when the voxel is embedded positionally, this value is used as an additional parameter to specify the weight of the voxel.

Only non-empty voxels are considered while embedding to reduce the computational time required. Removing the empty voxels creates a sequence of indefinite length, which can easily be processed using sequence-processing architectures like the Mamba-SSM.

For each voxel, we use absolute positional embedding with coordinates of the voxel grid and the count of points in that voxel to encode the data. Relative positional embedding or RoPE [27] can also be used for achieving higher accuracy, however, they tend to be slow as pairwise Euclidean distances need to be calculated.

This process is similar to KNN as both involve grouping points to reduce the input sequence lengths, however, KNN can go up to $O(N^2)$ in the worst case. The dimension of the voxel grid can be set according to the use case, density, and size of the input point cloud. The entire embedding process is depicted in Figure 1, where the input of dimensions $[B, N, 3]$ is transformed to $[B, N, 4]$. Optionally we use an MLP/1D-Convolution to augment the dimensions to $[B, N, D]$ before passing it to the main network.

D. State Space Models

S4 and Mamba models utilize a classical continuous system that processes a one-dimensional input function or sequence, $x(t) \in R$, through intermediate latent states $h(t) \in R^N$ to produce an output $y(t) \in R$. This process is represented by a linear Ordinary Differential Equation :

$$h(t) = Ah(t) + Bx(t) \tag{7}$$

$$y(t) = Ch(t) + Dx(t) \tag{8}$$

where $h(t) \in R^N$ is the hidden state. $h(t) \in R^N$ is the derivative of the hidden state. $A \in R^{N\times N}, B \in R^{N\times L}, C \in R^{L\times N}$, and $D \in R^{L\times L}$ are the parameters of the model, followed by a step to discretize the SSM using Zero Order Holding as :

$$A = e^{\Delta A} \tag{9}$$

$$B = \left(I - e^{\Delta A}\right) A^{-1} B \tag{10}$$

$$C = C \tag{11}$$

Therefore,

$$h_k = Ah_{k-1} + Bx_k \tag{12}$$

$$y_k = Ch_k \tag{13}$$

Final output y can be represented as the convolution of input x :

$$y = K * x \tag{14}$$

where K is the convolution kernel, which can be represented by:

TABLE I: Layer-wise architecture of VoxelMamba

Layer				**Depth-Index**	**Output Shape**	**Params**
MixerModel					[1,2771]	110,425,427
	T-Net			1-1	[1, 3, 10000]	133,614
	PointCloudEmbedding			1-2	[1, 2771, 1024]	143,168
	ModuleList			1-33		6,668,288
		Block		2-21	[1, 2771, 1024]	
			LayerNorm	3-1	[1, 2771, 1024]	2,048
			Mamba	3-2	[1, 2771, 1024]	6,666,240
	Identity			1-4		
	ModuleList			1-33		6,668,288
		Block		2-22	[1, 2771, 1024]	
			DropPath	3-3	[1, 2771, 1024]	
			LayerNorm	3-4	[1, 2771, 1024]	2,048
			Mamba	3-5	[1, 2771, 1024]	6,666,240
	Identity			1-6	[1, 2771, 1024]	
	Repeat xL					
	Identity			1-34	[1, 2771, 1024]	
	LayerNorm			1-35	[1, 2771, 1024]	2048
	Sequential			1-36	[1, 2771, 10]	2,788,580

$$K = \left(CB, CAB, \ldots, CA^{M-1}B\right) \tag{15}$$

Similar to an image, flattening the voxel grid would inevitably result in restricited receptive fields, as the relation between the voxels cannot be accurately estimated. Hence we use the bidirectional Mamba-SSM implementation provided in the Vision Mamba [5].

Mamba uses a Selective Scanning Mechanism means in which the matrices $B \in R^{B\times L\times N}, C \in R^{B\times N\times L}, \Delta \in R^{B\times L\times D}$ are derived from the input sequence $x \in R^{B\times L\times D}$ to learn the long-distance dependencies in the input sequence contextual information and ensures the dynamic range of weights within the model.

E. Main Network

In classification tasks, a classification token is appended to the middle of the position embedding. Experience has shown [5] that a token placed in the middle provides higher accuracy. The functions of $[B, N, D]$ are expanded using an initial MLP after position embedding is applied. The advanced functions are then routed through the primary network, which consists of a linear combination of MLP and the Mamba-Encoder block. This process is repeated L times, a hyper-parameter indicating the depth of the network. To increase the number of features and downsample the number of points between successive encoder layers, KNN is typically used. However, KNN consumes significant time, justifying the use of MLP to increase the dimensionality between successive encoder layers. The output header is a flexible module that can be adapted to the needs of segmentation and classification tasks. The Main network is depicted in Figure 2, and the layer-wise model architecture is shown in table 1.

IV. Experimental Analysis

A. Implementation details

Both segmentation and classification models are trained for 150 epochs with a batch size of 16 on an A100 colab GPU. A pre-trained version of the Transformation-Invariance module is used as a preprocessing step. The main network is trained using Adam optimizer with the learning rate set to 1e-5 with a cosine scheduler, momentum to 0.9, and weight decay to 0.05. The depth of the main network is set to 12, with a maximum pool in the output header for classification and MLP for segmentation. The iPhone 14's lidar sensor is used along with the Polycam app to create the custom dataset.

B. Classification

1) ModelNet40 Dataset: We use the ModelNet-40 dataset, which includes 12311 CAD models, divided into 9843 for training and 2468 for testing, to evaluate the classification capabilities of our models. To test the capabilities of the architecture with the Transformation-Invariance module, we expand the test dataset to 4936 by introducing random rotation, translation and scaling extensions. For the ModelNet40 dataset, the dimensions of the voxel grid are set to 25x25x25 as determined by experiments. The quantitative results of classification on the ModelNet40 dataset are shown in table 2, where H represents the dimensions of voxel grid used for classification. Notably, it can be seen from the table that the accuracy of our model depends on the granularity of the voxel grid. Since there is no relation between points that are grouped together in a single voxel, the accuracy of the model is less than other methods that consider some relation function between the points while grouping. Yet our model achieves near maximum state-of-the-art accuracy of 93.4%, while bearing the advantage of being computationally efficient.

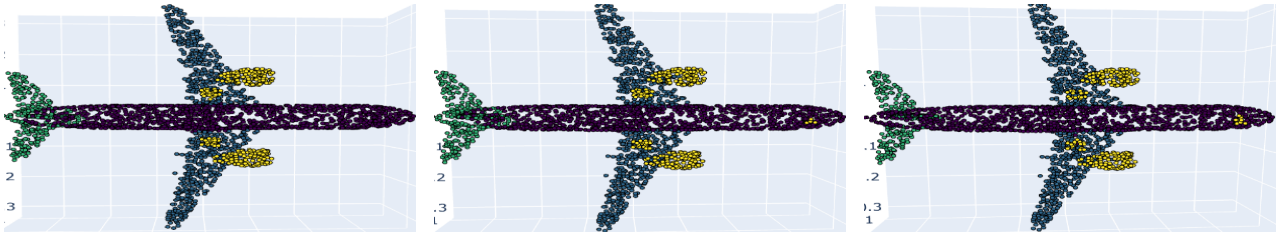
TABLE II: Comparison of point cloud classification accuracy on the ModelNet40 dataset. H represents dimensions of voxel grid

Model	**Accuracy**
VoxNet [11]	83%
MVCNN [9]	90.1%
binVoxNetPlus [28]	85.47%
OctFormer [26]	92.7%
PointNet++ [12]	91.9%
PointConv [1]	92.5%
PCT [3]	93.2%
PointMamba [24]	**93.6%**
Ours (H=10)	90.8%
Ours (H=20)	92.5%
Ours (H=100)	93.4%

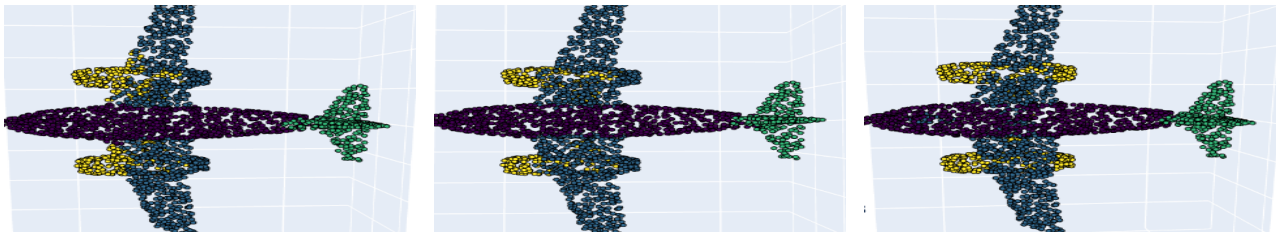
2) Custom Dataset: We also test our model on the custom created dataset. The dataset contains 108 point clouds of people riding two-wheelers. Hence we perform classification to classify whether a rider riding two wheeler exists, only two wheeler exists or nothing exists. The dataset contains 70 instances of a person riding two wheeler and 38 instances of only two-wheelers mixed with 100 random point clouds to generate a 208 point cloud dataset split into 156 train and 52 testing instances. A few sample point clouds are shown in Figure 4, and the results of classification are shown in table 3 below. Our model achieves 100% accuracy when detecting Rider+Two Wheeler. None of the samples from Rider+Two Wheeler or only Two Wheeler category are miss-classified as None category. Random Noise in Only Two Wheeler category causes it to be missclassified as Rider+Two Wheeler, and similarly noise in None category is sometimes miss-classified as one from other two categories, achieving a total accuracy of 89.5% and 93% accuracy in None and Only Two Wheeler categories.

C. Segmentation

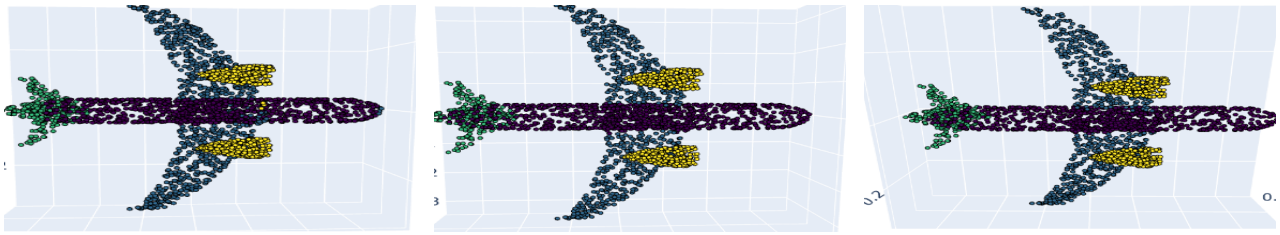
1) Airplane Part Segmentation: We perform part segmentation on an airplane dataset provided here . The dataset contains 2609 point clouds representing different airplanes. The qualitative comparison results with PointNet++ are presented in Figure 3, with both models being trained from scratch for



(a) PointNet++ (b) Proposed (c) Ground Truth



(d) PointNet++ (e) Proposed (f) Ground Truth



(g) PointNet++ (h) Proposed (i) Ground Truth

Fig. 3: Qualitative Segmentation Results on Airplane Part Dataset

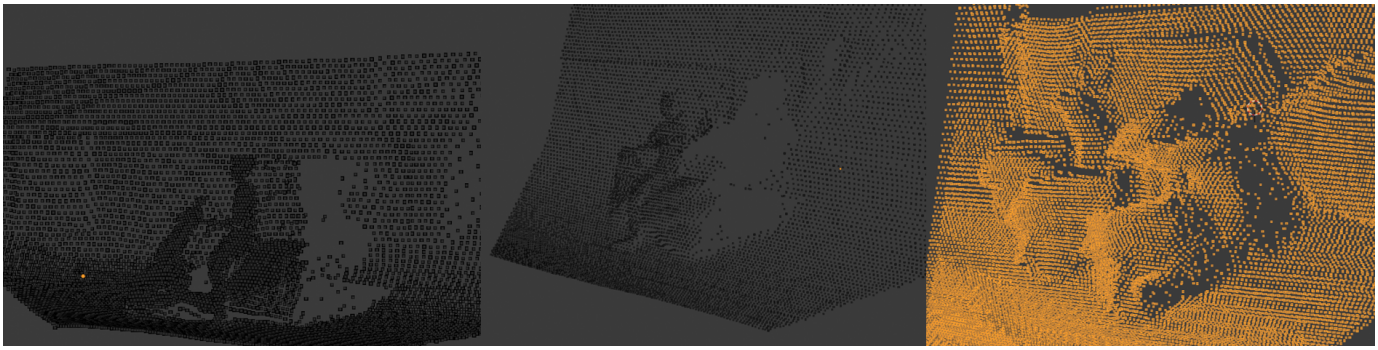


Fig. 4: Samples from custom Dataset

TABLE III: Classification results on custom dataset

Class	**Total**	**Detected**	**Accuracy**
Rider+TwoWheeler	70	70	100%
Two Wheeler	38	34	89.5%
None	100	93	93%

10 epochs. The model is trained from scratch and compared with PointNet++ [12] which is also trained from scratch on the same dataset. PointNet++ achieves an Overall Accuracy of 87.9% while our model achieves on Overall Accuracy of 92.6% on the validation dataset.

2) S3DIS Semantic Segmentation: The S3DIS is a indoor scene dataset for point cloud semantic segmentation. It contains 6 areas and 271 rooms. Each point in the dataset is divided into 13 categories. The quantitative segmentation results are provided in table 4. Our model easily surpasses CNNs, point and graph based methods, and closely competes with transformer architecture like PCT, achieving a mean Accuracy (mAcc) of 67.9 and mean Intersection over Union of 61.75.

We also compare the model complexity with other state-of-the-art models, shown in table 5. VoxelMamba contains 43.74M trainable parameters and 75.36G FLOPs at a network depth of 6 blocks. The number of trainable parameters is only dependent the network depth, while FLOPs is dependent both

TABLE IV: Comparison of the semantic segmentation performance on S3DIS dataset

Model	mAcc	mIoU
PointNet [2]	48.98	41.09
SEGCloud [29]	57.35	48.92
DGCNN [13]	84.10	56.10
PointCNN [30]	63.86	57.26
SPG [14]	66.50	58.04
PointWeb [31]	66.4	60.28
PCT [3]	67.65	61.33
Ours	**67.94**	**61.75**

on network depth and the length of input sequence.

TABLE V: Model complexity comparison

Model	Params (M)	FLOPs (G)
MVCNN	60.00	62.05
PointMamba	31.99	55.02
OctFormer	41.03	56.52
Ours (L=6)	43.74	75.36
Ours (L=16)	110.43	—

V. Ablation Study

To study the impact of individual architectural modules on the model's statistics, we perform a thorough ablation study on different parameters.

1) Voxel grid impact on point cloud density: It can be clearly seen from Figure 5 and 6 that output point cloud size follows a sigmoid-like curve as the grid size increases. The intial slow increase in the density can be linked to fact the point clouds are mostly empty, only concentrated in those regions where an object exists. Till the cell size becomes smaller than the object width, the number of voxels tend to remain the same. The slow increase at the end is also justified as the point cloud reaches saturation (No. of points = No. of Voxels), with each point being considered as a voxel.

2) Voxel grid impact on mathematical operations: Theoretically a linear increase in the number of mathematical operations is expected as the grid dimensions increase, but figure 7 suggests that its not quite the case. We use input point cloud that is randomly initialized, which is responsible for the sudden drop in the number of mathematical operations despite the increasing grid dimensions. This shows that the final sequence length to be processed by the Mamba-SSM module is highly dependent on the statistics of the input point cloud, such as the density, variance, standard deviation and mean.

3) Impact of model depth on runtime: Figure 8 shows the variation of model runtime w.r.t the variation in the network depth L (No. of Mamba blocks), when executed on colab A100 GPU. Again the sudden drops can be attributed to the random initialization, however approximating it as a straight line, it can be estimated that the at a depth of 20, the Mamba-SSM backbone consumes the same time as the entire process depicted in the figure 1 + Initial MLP/Feature Scaling.

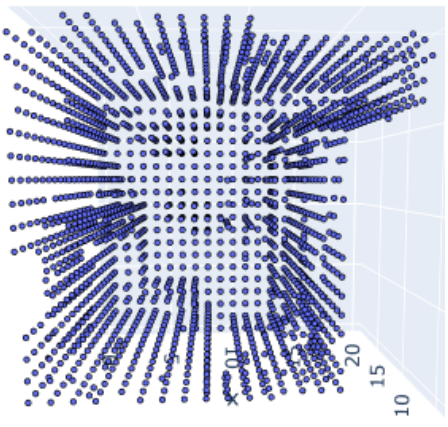
4) Impact of sequence length on resource consumption: Table 6 shows the variation in GPU memory usage when a fixed point cloud is passed through the network with different grid dimensions. This aids the hyperparameter setting when executing on a device of given specifications and limited memory. Furthermore, the low resource consumption as claimed is evident from the table when compared to PointMamba (Values for PointMamba are taken from official paper).

VI. Conclusion and Future Work

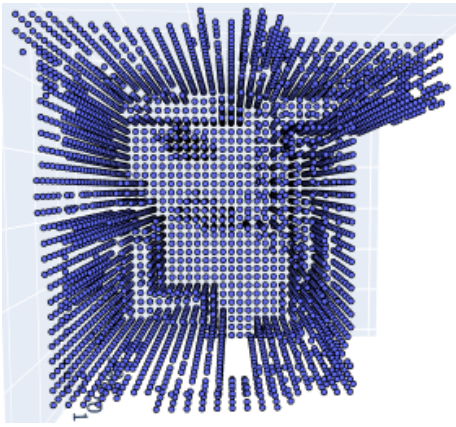
Hence in this paper, we propose a Mamba-SSM based point cloud processing architecture that uses voxelization as the embedding step. We also incorporate the Transformation-Invariance module to speed up the training process and increase the generalization capabilities of the model. To effectively capture features from non-casual data like point clouds,

Fig. 5: Visualization of Point Cloud with varying voxel grid size

(a) Point Cloud density at grid size 10x10x10



(b) Point Cloud density at grid size 30x30x40



(c) Point Cloud density at grid size 70x70x70

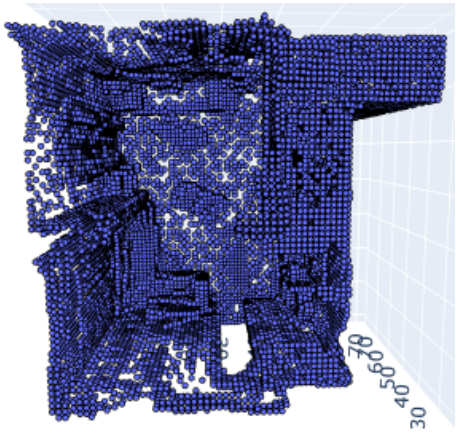


TABLE VI: Comparison of GPU Utilization at different configurations

GPU Usage (MB) (Ours)	**Sequence Lenght**	**Grid Dimensions**
2481	74	5
4661	1136	20
7545	8065	50
9369	10009	70
GPU Usage (MB) (PointMamba)		
∼8000	1024	-
∼20000	2048	-
∼35300	4096	-

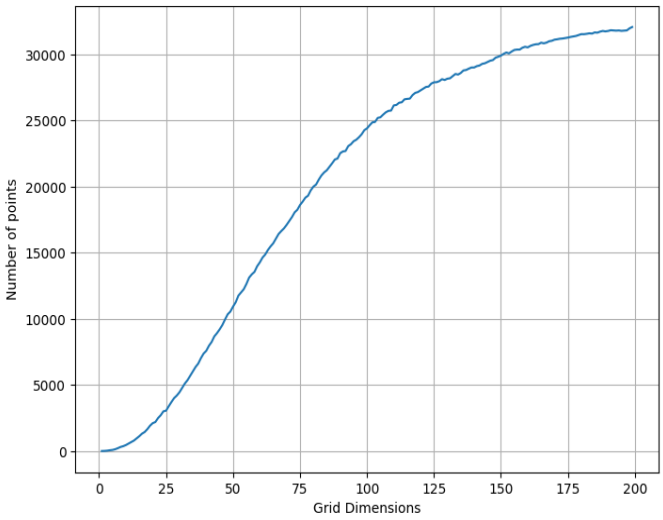


Fig. 6: Number of points after voxelization varying with grid dimensions

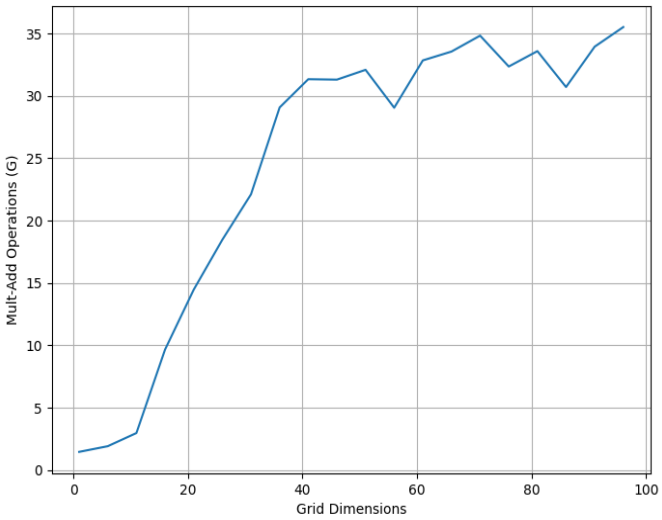


Fig. 7: Model complexity variation w.r.t grid dimensions

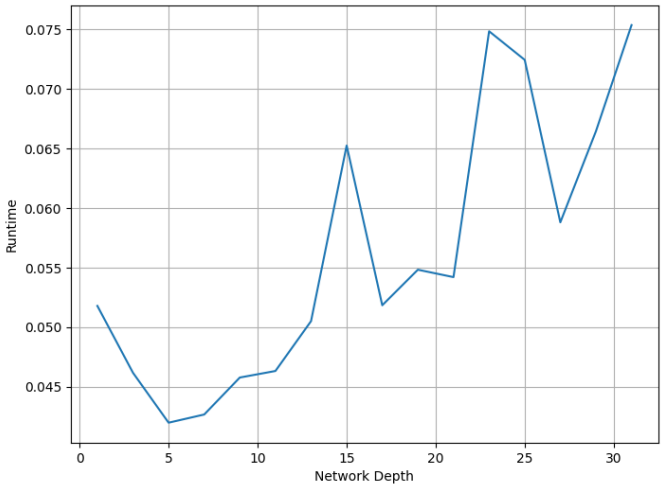


Fig. 8: Varation in runtime with the increase in network depth

we use bidirectional scanning method in the Mamba block. Experimental results show that proposed architecture achieves near best performance on classification and segmentation tasks, all the while scaling linearly w.r.t the time consumed. We also test the model on custom dataset achieving an average accuracy of 94.17% on classification. One potential improvement can be using a dynamic grid size for voxelization rather than a fixed one, where dimension are estimated according to the statistical distribution of the input point cloud.

REFERENCES

[1] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 9621–9630.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[3] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, pp. 187–199, 2021.

[4] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[5] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," *arXiv preprint arXiv:2401.09417*, 2024.

[6] Y. Liu, Y. Tian, Y. Zhao, H. Yu, L. Xie, Y. Wang, Q. Ye, and Y. Liu, "Vmamba: Visual state space model," *arXiv preprint arXiv:2401.10166*, 2024.

[7] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[8] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5010–5019.

[9] R. Feynman and F. Vernon, "Multi-view convolutional neural networks for 3d shape recognition," *Annals of physics*, vol. 24, pp. 118–173, 1963.

[10] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 2, 2021, pp. 1201–1209.

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[13] A. V. Phan, M. Le Nguyen, Y. L. H. Nguyen, and L. T. Bui, "Dgcnn: A convolutional neural network over large-scale labeled graphs," *Neural Networks*, vol. 108, pp. 533–543, 2018.

[14] Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov, "Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 446–15 456.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[18] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3164–3173.

[19] C. He, R. Li, S. Li, and L. Zhang, "Voxel set transformer: A set-to-set approach to 3d object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8417–8427.

[20] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler, faster, stronger," *arXiv preprint arXiv:2312.10035*, 2023.

[21] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems*, vol. 34, pp. 572–585, 2021.

[22] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[23] J. Liu, R. Yu, Y. Wang, Y. Zheng, T. Deng, W. Ye, and H. Wang, "Point mamba: A novel point cloud backbone based on state space model with octree-based ordering strategy," *arXiv preprint arXiv:2403.06467*, 2024.

[24] D. Liang, X. Zhou, X. Wang, X. Zhu, W. Xu, Z. Zou, X. Ye, and X. Bai, "Pointmamba: A simple state space model for point cloud analysis," 2024.

[25] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[26] P.-S. Wang, "Octformer: Octree-based transformers for 3d point clouds," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–11, 2023.

[27] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[28] C. Ma, Y. Guo, Y. Lei, and W. An, "Binary volumetric convolutional neural networks for 3-d object recognition," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 1, pp. 38–48, 2018.

[29] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *2017 international conference on 3D vision (3DV)*. IEEE, 2017, pp. 537–547.

[30] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, 2018.

[31] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5565–5573.