Efficient Point Cloud Analysis Using Hilbert Curve

Wanli Chen[1], Xinge Zhu[1], Guojin Chen[1], and Bei Yu[1]

The Chinese University of Hong Kong
{wlchen,gjchen21,byu}@cse.cuhk.edu.hk
zhuxinge123@gmail.com

Abstract. Some previous state-of-the-art research on analyzing point cloud rely on the voxelization quantization because it keeps the better spatial locality and geometry. However, these 3D voxelization methods and subsequent 3D convolution networks often bring the large computational overhead and GPU occupation. A straightforward alternative is to flatten 3D voxelization into 2D structure or utilize the pillar representation to perform the dimension reduction, while all of them would inevitably alter the spatial locality and 3D geometric information. In this way, we propose the HilbertNet to maintain the locality advantage of voxel-based methods while significantly reducing the computational cost. Here the key component is a new flattening mechanism based on Hilbert curve, which is a famous locality and geometry preserving function. Namely, if flattening 3D voxels using Hilbert curve encoding, the resulting structure will have similar spatial topology compared with original voxels. Through the Hilbert flattening, we can not only use 2D convolution (more lightweight than 3D convolution) to process voxels, but also incorporate technologies suitable in 2D space, such as transformer, to boost the performance. Our proposed HilbertNet achieves state-of-the-art performance on ShapeNet, ModelNet40 and S3DIS datasets with smaller cost and GPU occupation.

1 Introduction

Point clouds are the principal data form for 3D world; they also constitute the output of the 3D sensing device including LiDAR. Pioneering works often process point clouds directly, including PointNet [31], PointNet++ [34], SO-Net [19], etc. Point-wise methods bypass the mesh or voxelization reconstruction and possess the permutation invariance. However, in these methods the spatial locality of point clouds is not sufficiently attended. Recently, the success of deep convolutional networks for image processing has motivated the learning-based approach for point clouds and convolution is a natural function focusing on the spatial locality. One common methodology to manipulate the point cloud using convolutional neural networks is to first convert the raw point cloud into a volumetric representation, and then utilize 3D convolutional neural networks to extract the intermediate features. This approach can keep the spatial topology and generate well-generalized features; however, it usually introduces the huge computational

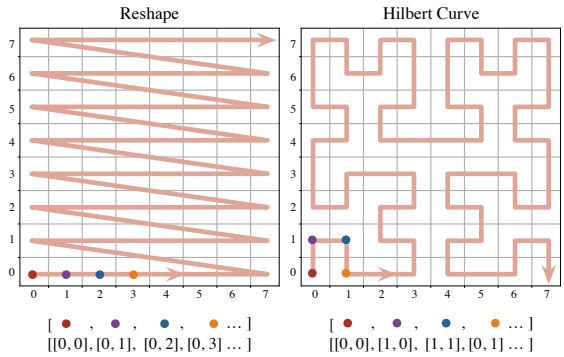


Fig. 1: Left: The mapping scheme of Reshape function. Right: The mapping scheme of Hilbert curve. Hilbert curve has better locality because it has no "**jump connections**" like reshape function.

cost and excessive memory usage (mainly from 3D convolutional neural networks).

A natural idea to tackle this issue is to map the 3D space to the 2D representation, including flattening, range image, pillar-based bird-eye view and multi-view image, etc, and then process the resulting 2D representation with 2D convolutional neural networks. Unfortunately, this approach would inevitably alter the spatial topology and locality. Hence, a locality preserving mapping function from three to two dimension is desired.

Here, space filling curves [27], widely existing in the database [2], GIS [36] and image compression [22], provide a possible solution, where they act as a **fractal** function going through each point in a multi-dimensional space without repetition (Note that fractal function is also applicable to efficient parallel implementation). This mechanism for linking all elements in the high-dimensional space reveals a way to reduce the dimension, namely, mapping elements in the high-dimensional space into low-dimensional space (i.e. 2D and 1D) according to the mapping rule of space filling curve. There are various space filling curves, including Z-order curve [28], Gray-Code [9] and Hilbert curve [14]. In this paper, because of the good locality preserving capability as shown in Fig. 1, Hilbert curve is incorporated to perform the dimension reduction and keep local topology.

Specifically, we first voxelize the point cloud to keep its 3D spatial locality. Then these voxels are flattened into 2D space via Hilbert curve in slice-level. We show an example in Equation (5), where for 2D compression, Hilbert curve stretches voxels slice by slice to get the 2D representation.

Then, in this work, we propose a local feature collection module Hilbert pooling, and a light-weighted global feature harvesting module called Hilbert attention. Besides it, we combine 2D features (obtained using 3D Hilbert compression) with 1D point features using Hilbert interpolation, which is designed for better 2D feature gathering. The cooperation of Hilbert curve based flattening and Hilbert curve based operations leads to the final framework, termed as Hilbert-

Net. We conduct extensive experiments on various tasks, including point cloud classification and segmentation, where it reaches state-of-the-art performance on ShapeNet [4], S3DIS [1] ModelNet40 [49] dataset while the GPU occupation and computational overhead are relatively small.

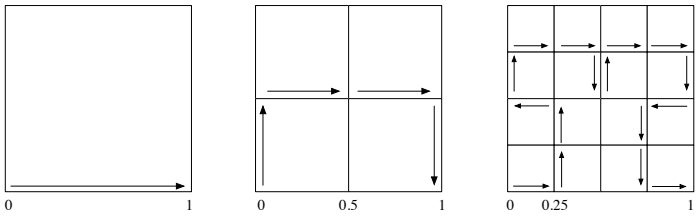
The contributions of this paper can be summarized as follows:

- We propose a new perspective of efficient usage of 3D volumetric data, namely, using the Hilbert curve to collapse 3D structure into 2D with spatial locality preserving and employing the 2D convolutions for lower overhead.
- We propose Hilbert interpolation and Hilbert pooling for better feature extraction. They are designed according to the characteristics of the Hilbert curve.
- We propose Hilbert attention, a light-weighted transformer based module for exploring the cross-slice attention, providing a solution for 2D global feature extraction.
- We design a powerful framework HilbertNet, which reaches SOTA performance on segmentation and classification benchmarks.

2 Related Work

Image-based Point Cloud Analysis. Recently, more and more 2D image-based methods are proposed for point cloud analysis. The advantages of using 2D images are 1) less computation than 3D convolution. 2) In 2D convolution, there are many excellent previous works [8, 10, 38, 48] that have been proved to have high generalization ability, which can be applied directly to point cloud tasks. However, it is not an easy task to use 2D convolution in the point cloud. An important step is to map the point cloud to 2D images. MVCNN [38] uses the snapshot of a point cloud in a different point of view to implement point-image mapping. This method cannot get accurate spatial information of a point cloud. Therefore, its performance on the segmentation task is not as good as the classification task. Another mapping method is using range image such as RangeNet++ [25] and SqueezeSeg [47]. Such methods make up for the shortcomings of image-based methods in segmentation, but in the process of 2D projection, the spatial locality of the original 3D structure is not well preserved. Different from previous works, PolarNet [56] adopts bird view and polar coordination to convert the point cloud into the 2D image and they achieve good results in the LiDAR point cloud segmentation task. In our proposed work, we use a new point-image mapping technique. The input image we use are voxels compressed by the 2D Hilbert curve. Such input images can ensure good spatial locality in each slice, so as to avoid information loss caused by point-image mapping.

Point Cloud with Space Filling Curve. Space filling curve [27] is a series of fractal curves that can fill the entire 2D/3D space. Classical space filling curve includes sweep curve, Z-order curve [28] and Hilbert curve [14], etc. The space-filling properties make them be extensively used in databases and GIS, and its



(a) (b) (c)

Fig. 2: Example of Hilbert curve generation.

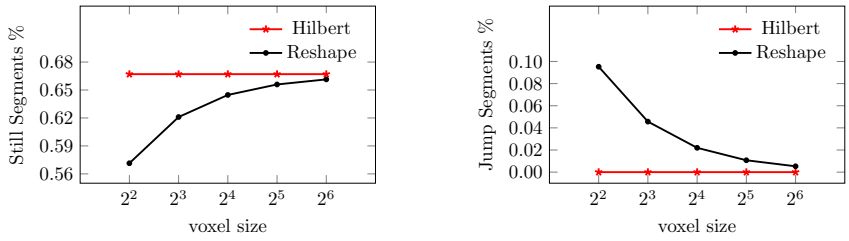
fractal feature also makes it applicable to parallel implementation. Similarly, such properties can be used to reduce the dimension of signals. In point cloud analysis, space filling curve usually appears with tree structure. For example, an early point cloud indexing work [41] used OCTree + 3D Hilbert curve to compress 3D point cloud information. Similarly, C-Flow [30] also describes point cloud using Hilbert curve. While O-CNN [42] uses OCTree + z-order curve to complete point cloud analysis with relatively low cost. Different space filling curves have different characteristics, and in this paper, we adopt the Hilbert curve because of its good locality preserving property.

Multi-View Fusion based Point Cloud Analysis. Point cloud can be naturally represented via point based, 2D image based and 3D voxel based methods [7,15,58,62]. The cooperation of different view representation might absorb the complementary advantages from each other and lead to a success to the high-performance point cloud analysis. For example, PVCNN [24] and Cylinder3D [60,63] adopt the voxel + point method (i.e. 1D and 3D combination) to balance the cost and accuracy. Image2Point [51], Volumetric MCVNN [32] adopt image-based method and voxel based method to perform the 2D and 3D view fusion. Different from previous work, in our proposed method, we use 1D sequence and 2D image as input, so as to incorporate the merit of multi-view fusion and obtain more friendly computational overhead and faster inference speed.

3 Methodologies

3.1 Hilbert Curve Preliminaries

Hilbert curve is a space filling curve that can link all elements in a space (as shown in Fig. 1), which often acts as a fractal function [35]. The formation of a 2D Hilbert curve is shown as follow. Firstly, we define $\mathfrak{I}$ and $\mathfrak{Q}$ as the interval of 1D space [0,1] and the starting point of Hilbert curve in 2D space $[0,1] \times [0,1]$ respectively. As shown in Fig. 2(a), for $z \in \mathbb{C}$, we first shrink z by $\frac{1}{2}$ along the origin, obtaining $z' = \frac{1}{2}z$. Then, we multiply a complex number i to rotate z' by 90°, obtaining $z'' = \frac{1}{2}zi$. Finally, we get the imaginary part of it: $z''' = -\bar{z}''$. We combine the three transformations and name them as $\mathfrak{N}_0$. The lower-left part of Fig. 2(b) is obtained by $\mathfrak{N}_0 z = \frac{1}{2}\bar{z}i$. Similarly, we shrink z by $\frac{1}{2}$ along starting point $\mathfrak{Q}$ and move it upward, obtaining upper-left part of Fig. 2(b), the



(a) Percentage of *Still* segments (b) Percentage of *Jump* segments

Fig. 3: Locality analysis of Hilbert curve and reshape function. Here we increase N from 2^2 to 2^6 and choose $D = 3$.

operation is $\mathfrak{N}_1 z = \frac{1}{2}z + \frac{i}{2}$. Next, we move upper-left by $\frac{1}{2}$ to get upper-right part of Fig. 2(a), the operation is $\mathfrak{N}_2 z = \frac{1}{2}z + \frac{i}{2} + \frac{1}{2}$. Finally, $\mathfrak{N}_3 z = -\frac{1}{2}\bar{z}i + 1 + \frac{i}{2}$ is obtained by rotating upper-left by -90° and move forward by 1. Because of the characteristics of Hilbert curve, it is more convenient to express it in quaternary form. Assuming $s \in \mathfrak{I}$, then: $s = 0.q_1q_2q_3...$ and $q_j = 0, 1, 2, 3$. The Hilbert curve is shown as the infinite combination of these transformations:

$$\mathcal{H}(s) = \begin{pmatrix} \mathcal{R}e \\ \mathcal{I}m \end{pmatrix} \lim_{n\to\infty} \mathfrak{N}_{q_1}\mathfrak{N}_{q_2}\mathfrak{N}_{q_3}...\mathfrak{N}_{q_n}\mathfrak{Q}. \tag{1}$$

Here n is the order of Hilbert curve. In real application, finite n-th order approximation is often used, which has the formation:

$$\mathcal{H}_n(s) = \mathcal{H}\left(0.q_1q_2q_3\ldots q_n\right) = \begin{pmatrix} \mathcal{R}e \\ \mathcal{I}m \end{pmatrix} \sum_{j=1}^{n} \left(\frac{1}{2^j}\right) H_{q_0}H_{q_1}H_{q_2}\ldots H_{q_{j-1}}h_{q_j};$$
$$H_0 z = \bar{z}i, H_1 z = z, H_2 z = z, H_3 z = -\bar{z}i; h_0 = 0, h_1 = i, h_2 = 1 + i, h_3 = 2 + i. \tag{2}$$

3.2 Advantages of Hilbert Curve

We claim Hilbert curve is more suitable for data flattening compared with PyTorch "reshape" function (as shown in Fig. 1) and we demonstrate this point in the perspective of locality preserving and structural similarity between flattened and original structure.

Advantage 1: Locality Preserving.
The locality of a space filling curve can be measured using the segments it contains [27]. The better locality, the better feature clustering property.

Definition 1 (Segments) *In space filling curve, the "line" that links two consecutive points is regarded as a segment. For a space filling curve, $N^D - 1$ segments appear to link N^D elements in a D-dimensional space with grid size N.*

Definition 2 (Jump Segments) *For two consecutive points P_i and P_{i+1} in a N-dimensional space, if the distance between them is larger than 1: $abs(P_{i+1} - P_i) > 1$. The segment that links P_i and P_{i+1} is regarded as Jump segment. Jump segment measures the number of jump connections.*

Definition 3 (Still Segments) *For two consecutive points P_i and P_{i+1} in a N-dimensional space, if the distance between P_i and P_{i+1} is equal to 0 in dimension k: $P_{i+1} = P_i$. The segment that links P_i and P_{i+1} is regarded as Still segment in dimension k. Still segment measures the extent that one space filling curve should move in one dimension to move in another dimension.*

The most appealing characteristic of the Hilbert curve is its good locality preserving property, which leads to better feature clustering property. In Fig. 1, we intuitively show its merit compared to reshape function. In the following, a theoretical explanation is given. The locality of a space filling curve can be measured using the percentage of *Jump* and *Still* segments. Smaller *Jump* segments mean smaller number of jump connections, and higher *Still* segments represent higher dimensional consistency. Both of them denote a better locality. The percentage of them are calculated using Equation (3).

$$\begin{aligned} J_R &= (\frac{N^D - 1}{N - 1} - D) \cdot \frac{1}{D(N^D - 1)}, \\ S_R &= (DN^D - N\frac{N^D - 1}{N - 1}) \cdot \frac{1}{D(N^D - 1)}, \\ J_H &= 0, \quad S_H = (D - 1)(N^D - 1) \cdot \frac{1}{D(N^D - 1)}, \end{aligned} \tag{3}$$

where J_R and S_R represent the *Jump* and *Still* segments of reshape function, and J_H and S_H represent the *Jump* and *Still* segments of Hilbert curve, respectively. N and D represent grid size and dimension. The comparison between reshape and Hilbert curve can be found in Fig. 3. It demonstrates that with the increasing of N, the percentage of *Still* segments of Hilbert curve is consistently larger than Reshape and the *Jump* segments of Hilbert curve are always zero. In this way, Hilbert curve is incorporated into our framework to perform the dimension reduction with good locality preserving.

Advantage 2: Lower Space to Linear Ratio
Besides the locality, the similarity between the original stricture and the flattened structure is also an important factor since flattening operations will inevitably alter the original structure which makes some continuous points (in original structure) distinct from each other after flattening. *space to linear ratio* (SLR) is then proposed to describe the similarity between the original structure and its flattened shape.

Definition 4 (Space to Linear Ratio) *If mapping a pair of points $p(t)$ and $p(\tau)$ in the 2D coordination in [0,1]×[0,1] to two points t and τ in 1D sequence in [0,1] using a space filling curve p, namely p:[0,1]→[0,1]×[0,1], the ratio*

$$\frac{|p(t) - p(\tau)|^2}{|t - \tau|} \tag{4}$$

is called space to linear ratio of the two points.

The upper bound of Equation (4) is called the space to linear ratio of the curve p. Obviously, the lower SLR, the more similarity between original structure and flattened structure, which also leads to better spatial locality.

Theorem 1 *The square-to-linear ratio of the Hilbert curve is equal to 6.*

The details of proof can be found in [3]. While for some consecutive points in PyTorch reshape function, the SLR is $4^n - 2^{n+1} + 2$ [59], where n is the curve order as defined before. It is obvious that Hilbert curve has lower SLR and therefore the flattened structure of Hilbert curve is closer to original one than PyTorch reshape function.

3.3 Pre-processing

All data are pre-processed via Voxelization and Hilbert Flattening Module (VHFM) before sending into neural network. Specifically, we first perform the uniform voxel partition [26, 32, 60], generating 3D data with size (R, R, R) and then these volumetric representations are flattened rapidly by applying Hilbert curve. Specifically, given a 3D feature $\mathcal{V}^{\in(C,R,R,R)}$ with channel size C, we first separate it into R slices along Z axis, where the slices $\mathcal{V}_{s1}, \mathcal{V}_{s2}...\mathcal{V}_{sR} \in (C, R, R, 1)$. Then, 2D n-th order Hilbert curve $\mathcal{H}_n(s)$ is used to encode each slice, obtaining R sequences with length R^2 as shown in Equation (5).

$$\mathcal{V}^{\in R\times R\times R} \rightarrow \begin{bmatrix} \mathcal{V}_{s1} \\ \mathcal{V}_{s2} \\ \vdots \\ \mathcal{V}_{sR} \end{bmatrix} \xrightarrow{\mathcal{H}_n(s)} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \vdots \\ \boldsymbol{s}_R \end{bmatrix} = \mathcal{I}. \tag{5}$$

The sequences $\boldsymbol{s}_1, \boldsymbol{s}_2...\boldsymbol{s}_R \in (C, R^2)$ and $\mathcal{H}_n(\boldsymbol{s}_k) = \mathcal{V}_{sk}, k = 1, 2...R$. After that we concatenate these sequences in Z axis order, obtaining 2D feature $\mathcal{I}^{\in(C,R^2,R)}$. Additional to the voxelization and Hilbert flattening, data augmentation such as rotation, flip and random jitting are applied to increase input diversity.

3.4 HilbertNet

Unlike previous point-voxel fusion based methods [24, 60], in our design, the voxels are flattened into 2D features for representing multi-view data. With the help of Hilbert curve, our method will not only lower the computational overhead but also preserve the 3D spatial locality. As shown in Fig. 4, HilbertNet acts as a two-branch network, to process 2D and 1D representation and perform the multi-view fusion. In the following, we will present every part of our method including Hilbert interpolation, Hilbert pooling, Hilbert attention in detail.

Feature Gathering. In our model, the 2D branch feature will be gathered in the form of point feature which is similar to the "devoxelization" process (*e.g.*

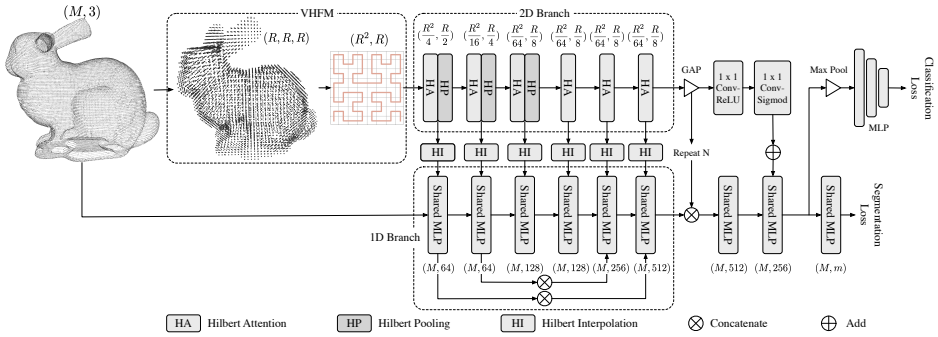


Fig. 4: The main framework of our model. Voxelization and Hilbert flattening module (VHFM) is the data processing part, where Hilbert curve is employed to perform the dimension reduction. Then a two-branch based network, HilbertNet, is designed to fuse 1D points and 2D slices representation to get final results. We employ global average pooling (GAP) and channel attention, which is comprehensively used in 2D tasks [16, 46].

linear interpolation [24], attention-weighted gathering [54]) proposed in previous works. Here we propose *Hilbert interpolation* for 2D feature gathering. To better demonstrate our advantage, we first introduce linear interpolation, a classical feature gathering method.

Linear Interpolation. Linear interpolation is a classical benchmark of voxel grid feature gathering method. Given a 3D feature $\mathcal{V}^{\in(C,R,R,R)}$, it is performed as:

$$\boldsymbol{O} = \text{Reshape}(\mathcal{V}) * \boldsymbol{F}_{linear}, \tag{6}$$

where $\boldsymbol{O}^{\in(M,C)}$ is the point representation with M sampling points and $\boldsymbol{F}_{linear}$ is the linear interpolation kernel (could be Bilinear or Trilinear), $*$ is convolution operation. Although it is widely used, linear interpolation has two problems. Firstly, due to the sparsity of the grid data, if applying linear interpolation, the addition of empty grids with non-empty grids will weaken the output non-empty part of feature. Secondly, the *Reshape*$(\cdot)$ function is not locality preserving, which also reduces the effectiveness of feature gathering.

Hilbert Interpolation. Given a 2D feature $\mathcal{I}^{\in(C,R^2,R)}$ that flattened by 3D feature and the target point cloud feature $\boldsymbol{O}^{\in(M,C)}$, The proposed Hilbert interpolation $\mathfrak{L}(\cdot)$ is performed as follow:

$$\begin{aligned} &\boldsymbol{O} = \mathfrak{L}(\mathcal{I}), \text{where} \\ &\mathcal{H}_{\lceil M \rceil}(\boldsymbol{O}) = \begin{cases} (\mathcal{I} \cdot \mathcal{W}_h) * \boldsymbol{F}_{linear}, & M \leq R^3; \\ \mathcal{I} * \boldsymbol{F}_{linear}, & M > R^3. \end{cases} \end{aligned} \tag{7}$$

Specifically, if the target size M is larger than the 2D feature size, we simply apply linear interpolation then reform the feature into $\boldsymbol{O}$ using Hilbert curve $\mathcal{H}_{\lceil M \rceil}(\cdot)$. $\lceil M \rceil$ represents the closest curve order that the corresponding Hilbert curve has at least M points. If $M \leq R^3$, $\mathcal{I}$ will multiply an adaptive weight

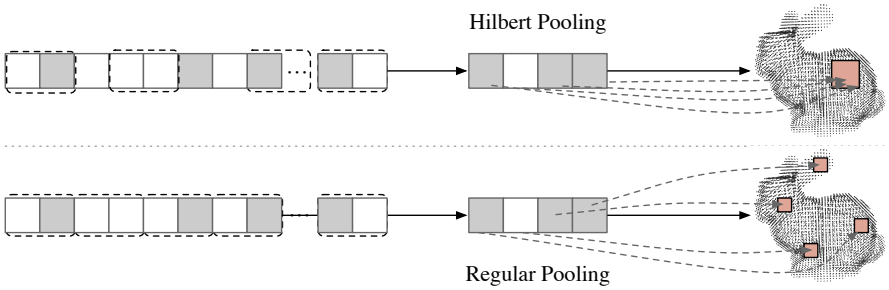


Fig. 5: Hilbert pooling and regular max pooling

$\mathcal{W}_h$, which is designed to emphasize the non-empty part in $\mathcal{I}$. Intuitively, $\mathcal{W}_h$ is the number of empty grids that fall in the linear interpolation kernel, the more empty grids, the larger compensation. To calculate $\mathcal{W}_h$, we first binarize featuremap $\mathcal{I}$ along channel C, obtaining $\mathcal{I}_B$. Then we apply sum filter to $\mathcal{I}_B$, obtaining $\mathcal{W}_B$:

$$\mathcal{W}_B = \mathcal{I}_B * \boldsymbol{F}_{sum}. \tag{8}$$

The kernel size of $\boldsymbol{F}_{sum}$ depends on the size of $\boldsymbol{F}_{linear}$ (marked as K_F) while the size of $\mathcal{W}_B$ is determined by the interpolation scale. Obviously, each element in $\mathcal{W}_B$ represents the number of non-empty grid that falls in the corresponding linear interpolation kernel. Finally, $\mathcal{W}_h^{\in(C,R^2,R)}$ is obtained by nearest expansion $\mathcal{N}(\cdot)$ of $\mathcal{W}_B$:

$$\mathcal{W}_h = K_F - \mathcal{N}(\mathcal{W}_B) + 1. \tag{9}$$

$\mathfrak{L}(\cdot)$ gathers 2D features using adaptive interpolation and Hilbert curve flattening, which overcomes the two problems in traditional linear interpolation. In the real implementation, we choose Bilinear interpolation kernel for $\mathfrak{L}(\cdot)$. After Hilbert interpolation, the gathered point feature $\boldsymbol{O}$ will merge with the 1D branch via additive fusion. The final output is formulated as:

$$\boldsymbol{Y} = \alpha(\boldsymbol{X}) + \boldsymbol{O}, \tag{10}$$

where $\boldsymbol{X}$ is point feature from 1D branch, $\alpha(\cdot)$ means shared MLP.

Hilbert Pooling. Different from previous 2D tasks, the 2D feature $\mathcal{I}$ in our model is obtained by the Hilbert curve, which means, if applying regular max pooling, the gathered feature follows the line of Hilbert curve instead of the original 3D structure. Therefore, the feature-to-be-pool may come from different parts of the original shape (See Fig. 5). Due to the specialty of $\mathcal{I}$, we design a novel pooling technique to harvest spatial information named Hilbert pooling $\mathfrak{P}(\cdot)$, specifically:

$$\text{MaxPool3D}(\mathcal{H}_n^{-1}(\mathcal{I})) \overset{\mathcal{H}_{n-1}(s)}{\longrightarrow} = \mathcal{I}' = \mathfrak{P}(\mathcal{I}), \tag{11}$$

where $\mathcal{H}_n^{-1}(\mathcal{I})$ is the inverse operation of Equation (5), which transform 2D feature into 3D. Then a 3D max pooling is applied to extract context information.

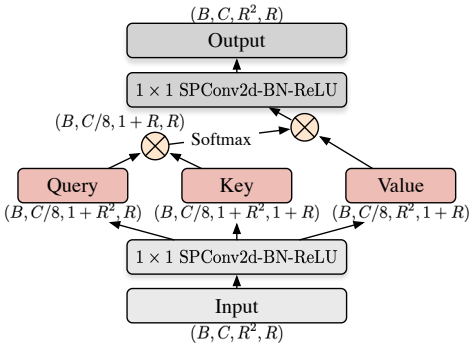


Fig. 6: Hilbert attention. It consists of 2D convolutions and matrix multiplications. Query and Key are generated using **4×1** convolution and **1×4** convolution respectively while Value features are extracted using **4×4** convolution. The shapes of $Query$, Key, $Value$ are asymmetric because of padding. Sparse convolution [11] is applied during experiments for lower GPU memory cost and faster inference speed.

After that, we apply $n-1$ order Hilbert curve $\mathcal{H}_{n-1}(s)$ to transform the resultant feature into 2D structure $\mathcal{I}'^{\in(C,\frac{R^2}{4},\frac{R}{2})}$ (similar to Equation (5)). It is noted that $\mathcal{H}_n(s) \approx \mathcal{H}_{n-1}(s)$ because Hilbert curve is a $fractal$ structure also, it has small SLR and good locality. These characteristics guarantee the distribution of features that before and after pooling are similar.

Hilbert Attention. In order to get the richer spatial feature in 2D branch, we introduce Self-attention [5, 44, 55], a powerful tool for global feature collection. Specifically, we propose Hilbert attention that focuses on space connection, which contains spatial information of all voxel grids. Hilbert attention is formed with 3 feature correlations: intra-slice correlation, inter-slice correlation, and mixed correlation. The detailed workflow is illustrated in Fig. 6.

Intra-Slice Correlation. VHFM module (see Equation (5)) transforms each slice $\mathcal{V}_{sk}$ into sequence $\boldsymbol{s}_k$, $k \in [1, R]$. Then, similar to [57] a pointwise linear projection $\sigma(\cdot)$ with weight w_{key} (marked as Key in Fig. 6):

$$\sigma(\mathcal{I}) = \sum_{e_k \in \boldsymbol{s}_k} w_{key} e_k \tag{12}$$

is applied along $\boldsymbol{s}_k$ for intra-slice level feature extraction, which collects pointwise feature **along** Hilbert curve.

Inter-Slice Correlation. To collect pointwise features between $\boldsymbol{s}_k$, we introduce inter-slice correlation. Specifically, the linear projection $\phi(\cdot)$ (marked as $Query$ in Fig. 6) is used:

$$\phi(\mathcal{I}) = \sigma(\mathcal{I}^\top), \tag{13}$$

where $\phi(\cdot)$ collects pointwise feature **across** Hilbert curve.

Mixed Correlation. Mixed Correlation $\gamma(\mathcal{I})$ considers spatial feature along and across Hilbert curve, which acts as a 4×4 convolution since the spatial neighbors in the 3D voxel is 8, the kernel size of convolution can be set as 4 to cover all

neighbors. Finally, Hilbert attention is gathered by considering the importance between inter-slice and intra-slice feature: $HA = \text{Softmax}(\phi(\mathcal{I})\sigma(\mathcal{I}))\gamma(\mathcal{I})$, where HA represents Hilbert attention.

4 Experiments

4.1 Implementation Details

Experimental Setting. HilbertNet is implemented based on PyTorch [29] framework and tested in the point cloud classification and part-segmentation task. For most of the experiments, we set Hilbert curve order $n = 6$, which means the chosen voxel size is 64^3. Adam optimizer [17] is applied during experiments with learning rate $lr = 0.001$ and batch size=16 for both tasks. The learning rate is reduced by half after every 50 epochs and the number of epoch is 200 for both part-segmentation and classification tasks.

Additionally, we conduct a large scale point cloud segmentation experiment on S3DIS [1] dataset with voxel size 128^3, the batch size in this experiment is set to 8 with 80 epochs for training, lr will be reduced by half after every 20 epochs. Other details are identical to the former tasks.

Classification Dataset. We evaluate the performance of our proposed model using ModelNet40 [49] dataset, which contains 9843 objects from 40 categories for training and 2468 objects for testing. Following the settings of previous work [19], we uniformly sampled 1024 points respectively during experiments.

Part-Segmentation Dataset. We use ShapeNetPart segmentation dataset [4] during experiments for part-segmentation tasks. It has 16 categories and 16881 objects in total. Each object in the dataset has 2 to 6 parts. Following previous works [24], we sample 2048 point clouds during experiments.

Large Scale Segmentation Dataset. S3DIS dataset [1] is used to test the performance of HilbertNet in large scale scene parsing task. S3DIS collects data from 271 rooms in 3 different building. Each point in the dataset is classified into 13 categories. Following a common procedure [54, 57], we apply Area 5 experiments.

4.2 Experimental Results

ModelNet40. For the ModelNet40 classification task, we add additional 3 FC layers after the network. The results of ModelNet40 can be found in Table 1. Our proposed model reaches SOTA performance with 1024 sample points similar to previous works [20,31,50]. Since our model aggregates 3D voxel information and 1D point information, the performance of our model is better than voxel-only methods and point-only methods such as VoxelNet [61] and PointNet [31].

ShapeNetPart. The overall performance of our model can be found in the right part of Table 1. Our model has the highest mIoU compared with previous

Table 1: Results on ModelNet40 & ShapeNetPart datasets

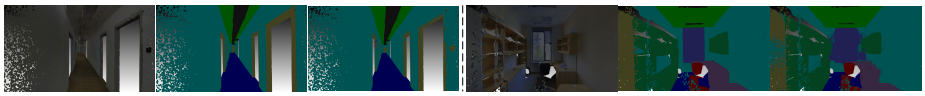
ModelNet40		ShapeNetPart	
Method	Acc	Method	mIoU
VoxNet [61]	85.9	Kd-Net [18]	82.3
Subvolume [33]	89.2	PointNet [31]	83.7
PointNet [31]	89.2	SO-Net [19]	84.9
DGCNN [45]	92.9	3D-GCN [23]	85.1
PointASNL [53]	92.9	DGCNN [45]	85.2
Grid-GCN [52]	93.1	PointCNN [21]	86.1
PCT [12]	93.2	PVCNN [24]	86.2
SO-Net [19]	93.4	KPConv [40]	86.4
CurveNet [50]	93.8	CurveNet [50]	86.6
Ours	**94.1**	Ours	**87.1**

Table 2: Comparison of methods

Method	voxel size	Inference time	mIoU
3D-UNet [26]	64^3	347ms	84.2
PVCNN [24]	32^3	62.5ms	86.0
HilbertNet-L	64^3	**42.1ms**	85.8
HilbertNet-M	64^3	59.2ms	86.4
HilbertNet	64^3	91.6ms	**87.1**

Table 3: Computational cost and GPU Memory of different methods. The tested voxel resolution is 32^3. (FLOPs: floating point operations)

Method	FLOPs	GPU Memory
3D Convolution	18.86G	162M
2D Convolution	4.45G	148.7M
Sparse 2D Convolution	1.47G	49.6M
NonLocal	0.34G	4G
Hilbert Attention	**0.32G**	**47.8M**



(a) Left to right: Point Cloud, GT, HilbertNet.

(b) Left to right: Point Cloud, GT, PointNet [31], KPConv [40], HilbertNet.

Fig. 7: (a) Visualized results on S3DIS Area 5 dataset; (b) Quantitative comparison.

works. With the rich location information from volumetric data, our model easily outperforms the point-only models such as PointNet [31], SO-Net [19]. Also, our model gets better results compared with other methods that use 2D or 3D features such as 3D-GCN [23].

S3DIS. We conduct S3DIS [1] Area 5 experiment to show that HilbertNet is qualified for large scale point cloud segmentation and results can be found in Table 4. Due to the successful combination of grid and point data, HilbertNet performs better than point-only [31], convolution-based [21] and transformer-based [57] methods. Some visualized results are posted in Fig. 7(a) while the visualized quantitative comparison between HilbertNet and other methods can be found in Fig. 7(b), in which HilbertNet shows better performance.

Inference Speed. Since our model is trained using both 3D and 1D data, we compare our model with a similar design such as PVCNN [24] for a fair comparison. Additionally, we propose HilbertNet-M (median) and HilbertNet-

Table 4: Results of S3DIS Area 5

	PointNet [31]	PointCNN [21]	PCCN [43]	MinkowskiNet [6]	KPConv [40]	PointTransformer [57]	HilbertNet
ceiling	88.8	92.3	92.3	91.8	92.8	94	**94.6**
floor	97.3	98.2	96.2	**98.7**	97.3	98.5	97.8
wall	69.8	79.4	75.9	86.2	82.4	86.3	**88.9**
beam	0.1	**0.3**	0	0	0	0	0
column	3.9	17.6	6	34.1	23.9	**38**	37.6
window	46.3	22.8	**69.5**	48.9	58	63.4	64.1
door	10.8	62.1	63.5	62.4	69	**74.3**	73.8
table	59	74.4	66.9	81.6	81.5	**89.1**	88.4
chair	52.6	80.6	65.6	89.8	**91**	82.4	85.4
sofa	5.9	31.7	47.3	47.2	**75.4**	74.3	73.5
bookcase	40.3	66.7	68.9	74.9	75.3	80.2	**82.7**
board	26.4	62.1	59.1	74.4	66.7	**76**	74.7
clutter	33.2	56.7	46.2	58.6	58.9	59.3	**60.1**
mIoU	41.1	57.3	58.3	65.4	67.1	70.4	**70.9**

L(light) during the experiment. HilbertNet-M has $0.5 \times C$ and HilbertNet-L has $0.25 \times C$, where C is the channel number of the features in HilbertNet. We also adopt a pure 3D model 3D-UNet [26] as the baseline and all the models are trained and tested on GTX TITAN X GPU. These models are trained using the identical setting as shown in their paper and ShapeNetPart is applied as our benchmark. The inference speed comparisons are listed in Table 2. It can be found that our model has a higher mIoU than PVCNN with comparable inference speed.

Next, we compare Hilbert attention with 3D convolution, 2D convolution, sparse 2D convolution and non-local attention in the perspective of FLOPs and GPU Memory usage. For a fair comparison, the kernel size for 2D and 3D convolution is set to 4 with 2 paddings. The result can be found in Table 3. It shows that Hilbert attention is more light-weighted than 3D convolution and also faster than 2D convolution and sparse 2D convolution. Moreover, Hilbert attention is much memory-saving than NonLocal attention.

4.3 Ablation Study

In this part, we conduct a set of ablation studies in the perspective of the flattening method, gathering method, pooling method, and convolution method to evaluate the influence of each module in our model. The ModelNet40 dataset is used during the ablation study.

Hilbert Curve vs. Reshape Function. Our design is based on the Hilbert curve, due to its better locality preserving property compared with other space filling curves such as reshape function. As shown in column 1 and column 6 in Fig. 8, we make a comparison to evaluate different flattening methods. Here we simply replace all the Hilbert curve $\mathcal{H}_n(s)$ in HilbertNet with reshape function including Hilbert interpolation, Hilbert pooling, and Hilbert attention. It can be

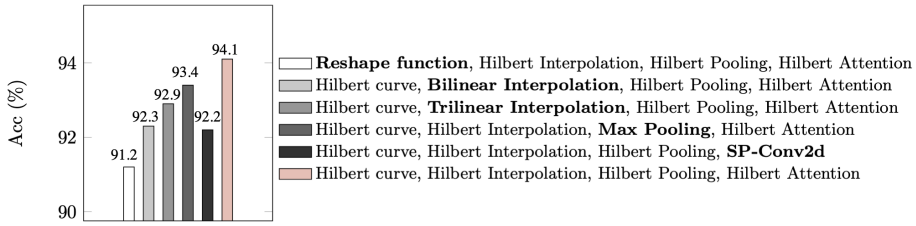


Fig. 8: Ablation study results of HilbertNet. The label refers to: **Flattening Method, Gathering Method, Pooling Method, Convolution Method**.

found that the Hilbert curve achieves the higher accuracy (94.1% vs. 91.2 %), which further illustrates its advantage over reshape function.

Hilbert Interpolation vs. Linear Interpolation. In this experiment, we conduct Bilinear interpolation, Trilinear interpolation (see Equation (6)) and Hilbert interpolation to test the performance of these voxel feature gathering methods. The results in columns 2, 3, 6 of Fig. 8 show that Hilbert interpolation has an obvious advantage over the other methods. This is because Hilbert interpolation not only applies Hilbert curve, which has a better locality but also enhances the non-empty part in grid data.

Hilbert Pooling vs. 2D Max Pooling. Max pooling is a common technique for 2D feature downsizing and it has been widely used in 2D convolutional networks [13,37,39]. However, the 2D featuremap $\mathcal{I}$ in our design is obtained via 3D flattening, which makes 2D max pooling not feasible for HilbertNet. Therefore we design Hilbert pooling and we compare the performance of it with regular max pooling. The results in columns 4 and 6 of Fig. 8 demonstrate that our proposed Hilbert pooling module is more suitable for handling data that is flattened by Hilbert curve.

Hilbert Attention vs. 2D Convolution. In this part, We simply replace the Hilbert attention with 4×4 sparse 2D convolution. The result can be found in columns 5 and 6 of Fig. 8. This result demonstrates that Hilbert attention plays a crucial role in exploring spatial information across 2D flattened voxels.

5 Conclusion

In this paper, we propose a novel framework for efficient and effective point cloud analysis, that is, introducing Hilbert curve to reduce the dimension of volumetric data, preserving spatial locality and topology, and using 2D convolutions for processing, bypassing the cumbersome 3D convolutions. Based on the proposed Hilbert curve flattening methods, we design the two-branch based HilbertNet, which copes with 1D sequences and 2D slices respectively, and fuses these two views together. Additionally, we propose two useful local feature harvesting modules and one light-weighted attention for exploring the cross-slice context information. Our proposed method is proved to be efficient and effective in point cloud classification and segmentation tasks.

References

1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1534–1543 (2016) 3, 11, 12
2. Balkić, Z., Šoštarić, D., Horvat, G.: Geohash and uuid identifier for multi-agent systems. In: KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications. pp. 290–298. Springer (2012) 2
3. Bauman, K.E.: The dilation factor of the peano-hilbert curve. Mathematical Notes **80**(5), 609–620 (2006) 7
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 3, 11
5. Chen, Y., Kalantidis, Y., Li, J., Yan, S., Feng, J.: Aˆ 2-Nets: Double attention networks. In: Proc. NIPS. pp. 352–361 (2018) 10
6. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019) 13
7. Cong, P., Zhu, X., Ma, Y.: Input-output balanced framework for long-tailed lidar semantic segmentation. 2021 IEEE International Conference on Multimedia and Expo (ICME) pp. 1–6 (2021) 4
8. Esteves, C., Xu, Y., Allen-Blanchette, C., Daniilidis, K.: Equivariant multi-view networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1568–1577 (2019) 3
9. Faloutsos, C.: Multiattribute hashing using gray codes. In: Proceedings of the 1986 ACM SIGMOD international conference on Management of data. pp. 227–238 (1986) 2
10. Feng, Y., Zhang, Z., Zhao, X., Ji, R., Gao, Y.: Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 264–272 (2018) 3
11. Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307 (2017) 10
12. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. arXiv preprint arXiv:2012.09688 (2020) 12
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR. pp. 770–778 (2016) 14
14. Hilbert, D.: Über die stetige abbildung einer linie auf ein flächenstück. In: Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes, pp. 1–2. Springer (1935) 2, 3
15. Hou, Y., Zhu, X., Ma, Y., Loy, C.C., Li, Y.: Point-to-voxel knowledge distillation for lidar semantic segmentation. ArXiv **abs/2206.02099** (2022) 4
16. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. CVPR. pp. 7132–7141 (2018) 8
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. ICLR (2015) 11
18. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: ICCV. pp. 863–872 (2017) 12
19. Li, J., Chen, B.M., Lee, G.H.: SO-Net: Self-organizing network for point cloud analysis. In: Proc. CVPR (2018) 1, 11, 12

20. Li, J., Chen, B.M., Lee, G.H.: So-net: Self-organizing network for point cloud analysis. In: CVPR. pp. 9397–9406 (2018) 11
21. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on X-transformed points. In: Proc. NIPS, pp. 820–830 (2018) 12, 13
22. Liang, J.Y., Chen, C.S., Huang, C.H., Liu, L.: Lossless compression of medical images using hilbert space-filling curves. Computerized Medical Imaging and Graphics **32**(3), 174–182 (2008) 2
23. Lin, Z.H., Huang, S.Y., Wang, Y.C.F.: Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In: CVPR. pp. 1800–1809 (2020) 12
24. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. arXiv preprint arXiv:1907.03739 (2019) 4, 7, 8, 11, 12
25. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: IROS. pp. 4213–4220. IEEE (2019) 3
26. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 2016 fourth international conference on 3D vision (3DV). pp. 565–571. IEEE (2016) 7, 12, 13
27. Mokbel, M.F., Aref, W.G., Kamel, I.: Analysis of multi-dimensional space-filling curves. GeoInformatica **7**(3), 179–209 (2003) 2, 3, 5
28. Orenstein, J.A.: Spatial query processing in an object-oriented database system. In: Proceedings of the 1986 ACM SIGMOD international conference on Management of data. pp. 326–336 (1986) 2, 3
29. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Workshop (2017) 11
30. Pumarola, A., Popov, S., Moreno-Noguer, F., Ferrari, V.: C-flow: Conditional generative flow models for images and 3d point clouds. In: CVPR. pp. 7949–7958 (2020) 4
31. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017) 1, 11, 12, 13
32. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5648–5656 (2016) 4, 7
33. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: CVPR. pp. 5648–5656 (2016) 12
34. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017) 1
35. Sagan, H.: Space-filling curves. Springer Science & Business Media (2012) 4
36. Samet, H.: Applications of spatial data structures: computer graphics, image processing, and GIS. Addison-Wesley Longman Publishing Co., Inc. (1990) 2
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. ICLR. pp. 1–14 (2015) 14
38. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV. pp. 945–953 (2015) 3
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. CVPR. pp. 1–9 (2015) 14
40. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV. pp. 6411–6420 (2019) 12, 13

41. Wang, J., Shan, J.: Space filling curve based point clouds index. In: Proceedings of the 8th International Conference on GeoComputation. pp. 551–562. Citeseer (2005) 4
42. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017) 4
43. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2589–2597 (2018) 13
44. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proc. CVPR. pp. 7794–7803 (2018) 10
45. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. TOG **38**(5), 1–12 (2019) 12
46. Woo, S., Park, J., Lee, J.Y., So Kweon, I.: CBAM: Convolutional block attention module. In: Proc. ECCV. pp. 3–19 (2018) 8
47. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In: ICRA. pp. 1887–1893. IEEE (2018) 3
48. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR. pp. 9621–9630 (2019) 3
49. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR. pp. 1912–1920 (2015) 3, 11
50. Xiang, T., Zhang, C., Song, Y., Yu, J., Cai, W.: Walk in the cloud: Learning curves for point clouds shape analysis. arXiv preprint arXiv:2105.01288 (2021) 11, 12
51. Xu, C., Yang, S., Zhai, B., Wu, B., Yue, X., Zhan, W., Vajda, P., Keutzer, K., Tomizuka, M.: Image2point: 3d point-cloud understanding with pretrained 2d convnets. arXiv preprint arXiv:2106.04180 (2021) 4
52. Xu, Q., Sun, X., Wu, C.Y., Wang, P., Neumann, U.: Grid-gcn for fast and scalable point cloud learning. In: CVPR. pp. 5661–5670 (2020) 12
53. Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S.: Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In: CVPR. pp. 5589–5598 (2020) 12
54. Ye, M., Xu, S., Cao, T., Chen, Q.: Drinet: A dual-representation iterative learning network for point cloud segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7447–7456 (2021) 8, 11
55. Zhang, S., He, X., Yan, S.: LatentGNN: Learning efficient non-local relations for visual recognition. In: Proc. ICML. pp. 7374–7383 (2019) 10
56. Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In: CVPR. pp. 9601–9610 (2020) 3
57. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021) 10, 11, 12, 13
58. Zhao, L., Zhou, H., Zhu, X., Song, X., Li, H., Tao, W.: Lif-seg: Lidar and camera image fusion for 3d lidar semantic segmentation. ArXiv **abs/2108.07511** (2021) 4
59. Zhao, Q., Zhou, Z., Dou, S., Li, Y., Lu, R., Wang, Y., Zhao, C.: Rethinking the zigzag flattening for image reading. arXiv preprint arXiv:2202.10240 (2022) 7

60. Zhou, H., Zhu, X., Song, X., Ma, Y., Wang, Z., Li, H., Lin, D.: Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. arXiv preprint arXiv:2008.01550 (2020) 4, 7
61. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR. pp. 4490–4499 (2018) 11, 12
62. Zhu, X., Zhou, H., Wang, T., Hong, F., Li, W., Ma, Y., Li, H., Yang, R., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar-based perception. IEEE transactions on pattern analysis and machine intelligence **PP** (2021) 4
63. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. CVPR pp. 9934–9943 (2021) 4