The Journal of Supercomputing (2022) 78:1479–1500
https://doi.org/10.1007/s11227-021-03899-x

A voxelized point clouds representation for object classification and segmentation on 3D data

Abubakar Sulaiman Gezawa[1] · Zikirillahi A. Bello[1] · Qicong Wang[1,2] · Lei Yunqi[1]

Accepted: 15 May 2021 / Published online: 10 June 2021

Abstract

Processing large amount of high-resolution 3D data requires enormous computational resources. As a result, a suitable 3D data representation must be chosen, and the data must be simplified to a size that can be easily processed. The question is how can the data be simplified? Random point sampling is a common sampling strategy. However, it is sensitive to changes in density. We build a sampling module based on a hybrid model that combines point cloud and voxel data. To determine the relationship between points within each voxel, the module uses the magnitude of the point (the Euclidean distance between the point and the object's center) along with angles between each point embedded within each voxel. By exploiting farthest point sampling (FPS) that begins with a point in the set and selects the farthest point from the points already selected iteratively, our method has the advantage of covering the whole point set within a given number of centroids and still maintains the key benefits of both point cloud and voxel to better characterize geometric details contains in a 3D shape. With further observation that the number of points in each cell differs, we use a point quantization method to ensure that each cell has the same number of points. This allows all voxels to have the same feature size vector, making it easier for 3D convolution kernels to extract object features. We demonstrate these benefits and make comparisons with solid baselines on ModelNet10, ModelNet40 and ShapeNetPart datasets, demonstrating that our method outperforms some deep learning approaches for shape classification and segmentation tasks.

Keywords Point sampling · Object classification · Shape segmentation · 3D data representation

✉ Lei Yunqi
yqlei@xmu.edu.cn

Extended author information available on the last page of the article

1 Introduction

In recent years, capturing 3D data has become much easier. Point clouds, multi-view images, and voxel grids are some examples of how this data can be represented. Comprehensive survey for 3D data representations is found in [1]. In the field of computer vision, image feature extraction is commonly done using convolutional neural networks (CNNs) and in most image processing and understanding activities, they have proved to be superior to handcrafted solutions. Adapting a CNN designed for frequently sampled 2D images, on the other hand, is a difficult challenge to irregular triangle meshes or point clouds as models for 3D shapes. A point cloud is a collection of data about a 3D object's surface, although a grid-based representation often includes information about available space, the former is much more effective. Processing a point cloud, however, can be difficult because it may contain a large number of data points in it. In certain ways, lowering the number of points can be advantageous, for example, power consumption reduction, computational cost reduction, and communication load reduction, etc.

In recent years, there has been a significant increase in deep learning methods been used to analyze point cloud data with great success. A point cloud can be used for a variety of shape analysis tasks, such as classification [2–8], component segmentation [2–4, 7–9], semantic segmentation [2, 4, 7, 10–12] and more. Volumetric grid on the other hand such as VoxNet and its improvement [7, 13–17] are the simplest methods for converting a three-dimensional model into an occupancy grid. Although, a simple VoxNet implementation have scalability problems because the computational cost expand cubically with the 3D grid resolution for dense 3D data. The meanings of numerous abbreviations and acronyms used throughout the paper are specified in Table 1, along with the page where each is defined or first used.

The key downside of the volumetric method is information loss due to voxelization and huge computational cost as the resolution of the voxel increase. The aim of Kd-Net [18] and Octree-Nets [19, 20] is to solve these drawbacks by leaving out computations on empty cells and concentrating on informative ones. These networks, however, are difficult to effectively implement. Methods based on manifold [21, 22] compute CNN over features specified on a 3D mesh manifold. These methods work best with smooth manifold surfaces and are vulnerable to noise and large distortion, as a result, they are unsuitable for non-manifold 3D models in many datasets. Approaches that rely on multi-view images [23–25] convert the 3D shape into a sequence of 2D images taken from various angles and feed the CNN with the stacked images. Although, it is not clear how to work out the view positions to prevent self-occlusion by covering all 3D shapes.

We propose a hybrid network that incorporates point cloud and voxel grid data representations to optimize the benefits of each. Our network receives a point cloud embedded in a volumetric grid structure. We are motivated by the performance shown by point cloud and voxels in 3D shape analysis tasks. Randomly sampling a subset of points is one naive approach to reducing the data load. This method of sampling, in addition to other methods found in the literature [26, 27],

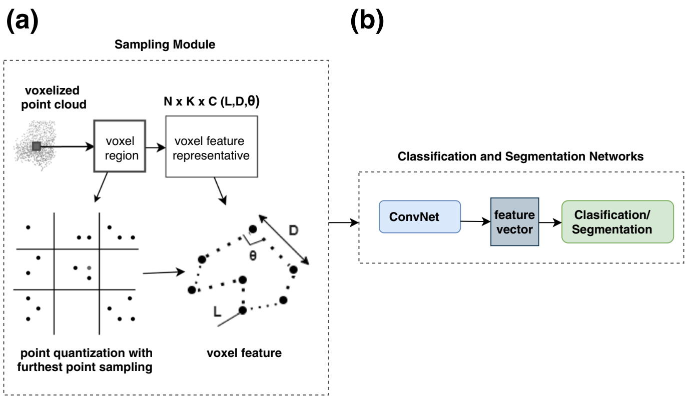
(a)
(b)
Sampling Module
voxelized point cloud
N x K x C (L,D,θ)
voxel region
voxel feature representative
point quantization with furthest point sampling
voxel feature
D
θ
L
Classification and Segmentation Networks
ConvNet
feature vector
Clasification/ Segmentation

Fig. 1 Overview of our Model. **a** Sampling module. The input is a voxelized point cloud representation of the 3D shape, after which a point quantization is performed to quantify all points within each cell, and if there are more than K points, a furthest point sampling is used to sample K of them. N is the size of voxel; K is the number of points sampled per voxel, c is the size of voxel features. For each voxel, these three variables are calculated and stacked as features. (b) Classification and Segmentation networks

Table 1 The table below defines the meaning of various abbreviations and acronyms used throughout the paper. The page where each is defined or first used is also provided

Abbreviation	Meaning	Page
CNNs	Convolutional neural networks	2
RPS	Random point sampling	3
FPS	Farthest point sampling	3
SVM	Support-vector machine	4
KNN	k-Nearest neighbor	4
WKS	Wave kernel signatures	4
HKS	Heat kernel signatures	4
BoF	Bag of features	4
DBN	Deep belief network	4
GAN	Generative adversarial network	4
RCC	Reflection convolution concatenation	5
SOM	Self organizing map	6
SIFT	Scale-invariant feature transform	6
mIoU	Mean intersection over union	14

does not create a simplified point cloud that is better suited to a later task like segmentation, classification and many others. Two opposing limitations must be reconciled in the condensed point cloud. On one side, there is a needs maintain resemblance to the original shape. However, we want to improve it for a future task. We overcome this problem by using farthest point sampling (FPS), which has the advantage of sampling only a subset of the original points. Its goal is to cover as much of the input as possible. Figure 1 shows the model overview of our method.

The main technical problem is that the number of points in each cell varies as a result; we use a point quantization method to ensure that each cell has the same number of points. This makes it simpler for 3D convolution kernels to extract object features because all voxels have the same feature size vector. We used a single module to extract the features of the voxel that serve as input to our network. Our method's robustness in 3D form classification and segmentation tasks has been demonstrated by experiments on standard 3D datasets.

Our main contributions are given as follows:

- We present a hybrid 3D data representations approach that improve the local geometric details of a 3D object by utilizing point cloud and voxels
- We construct a sampling module that uses the magnitude of the point (the Euclidean distance between the point and the object's center), as well as the distances and angles between each point embedded within each voxel, to determine the relationship between points within each voxel that are stacked together as features.
- FPS was used to sample points within each voxel, and then a point quantization approach was used to ensure a constant number of points in each cell, allowing all voxels to share the same feature size vector, making 3D convolution kernels to extract object features easier.
- Enhanced performance for classification and segmentation tasks with sample point clouds in contrast to other sampling alternatives

2 Literature reviewed

We begin by reviewing existing handcrafted features and other deep learning methods for 3D shape analysis in this section. Then, we discuss the point cloud simplification and sampling methods used in a variety of graphic applications.

2.1 Handcrafted features

Many machine learning approaches generate 3D descriptors by extracting lower-level features from data and feeding these features to the model to generate 3D descriptors. Some of this are geometric features consist of Gaussian curvature and mean curvature [28], average geodesic distance [29], spin images [30]. Recent spatial features such as wave kernel signatures (WKS), heat kernel signatures (HKS), and other heat-based signatures have also been used in the literature for local feature extraction [31–33]. On these features, some methods use machine learning techniques directly (e.g., random forest, support-vector machine (SVM), k-nearest neighbor (kNN) [34], correspondence study [35] or use some greedy and local processes, such as k means [36], region growing [37]). Kazmi et al. [38] provide a detailed analysis on 2D and 3D descriptors.

The majority of previous reviews, on the other hand, have concentrated on conventional methods for generating 3D shape descriptors. Rostami et al. [39] recently published a thorough analysis on data-driven 3D Shape descriptors. In this study, the 3D descriptors are divided into two main categories which are shallow descriptors and deep shape descriptors. The shallow descriptors are subdivided further into optimization-based descriptors, which are often implemented in a supervised manner [30] and clustering-based descriptors that are mostly unsupervised and are built using bag of features technique (BoF) [40]. The deep shape descriptors are subdivided into probabilistic models [41], auto-encoding [42], or CNN [43]. The probabilistic groups are again sub-divided into deep belief network (DBN) based and generative adversarial network (GAN) based. Deep learning models had the advantage of being able to efficiently learn hierarchical discriminative features.

2.2 Deep learning

For 3D shape analysis, a set of deep learning methods has been presented. According to the 3D shape representation used in each solution, we divide these approaches into several categories.

2.2.1 Voxel based methods

The full geometry of the models is used in these approaches. [13] Proposes 3D shapeNets, which transform input objects into a binary tensor of 30x30x30 dimensions. Despite the method's good efficiency, it has a number of limitations, such as adding more dimension to the convolutional kernel, which makes processing high resolution inputs more difficult. With less input parameters, Voxnet[14] improved [13], but it is still limited to low resolution due to the computational expense. Some techniques use the sparse voxel representation for 3D objects [20, 44–46] and perform network operations on the octree data structure similar to [47], but the complexity of these network structures is one of the major disadvantages of these methods. LightNet is a real-time volumetric CNN that was proposed by [48]for 3D object recognition tasks. The network architecture has two major capabilities: it can learn a large number of features at once using multi-tasking, and it can achieve quick convergence with fewer parameters by combining the activation and convolution operations with the batch normalization process. For classifications tasks, the network outperformed [14] by above 23% in both ModelNet10 and ModeleNe40 datasets.

NormalNet is a voxel-based CNN proposed by [49] for 3D shape retrieval and classification tasks. Instead of binary voxels, normal vectors of the object surfaces are used as input in this process. The authors propose a reflection convolution concatenation (RCC) module for extracting simple features for 3D vision tasks while keeping the number of parameters to a minimum. On the ModelNet10 and ModelNet40 datasets, the network performs well in 3D shape retrieval and classification tasks. Despite the fact that volumetric 3D models are efficient, most current architectures require a considerable amount of computational resources due to the convolution process and the large number of parameters.

2.2.2 Multi view based methods

These methods generate a large number of images from a variety of perspectives, which are then fed into a 2D CNN [23–25, 50]. Kanezakiet al. [51] proposed RotationNet, which takes as input multi-view images of an object and estimates both the pose and the object type. Unlike previous methods that trained using known view point labels, this approach treats view point labels as latent variables. For inference, the network only uses a subset of multi-view images. Feng et al. [52] propose group view CNN (GVCNN) to exploit the inherent hierarchical connection and discrimination among views, in contrast to the view to shape setting commonly used by many methods. This model is made up of a hierarchical view group shape architecture that is divided into three levels: view, group, and shape, all of which can be rearranged using a grouping strategy. On the ModelNet40 dataset, this method performed well on 3D shape classification tasks.

Despite the fact that these methods can directly exploit image-based CNNs for 3D shape analysis and handle high-resolution inputs, it is uncertain how to figure out how many views to have and how to distribute them to fill the 3D shape while preventing self-occlusions. Our approach is based on a hybrid 3D data representation that eliminates the need for view selection. It can also manage high-resolution inputs and produce results that are comparable to multi-view-based approaches in terms of efficiency and accuracy.

2.2.3 Manifold based methods

Many of these approaches use CNN operations on a 3D mesh manifold's geometric features. Some methods convert 3D surfaces to 2D patches and then parameterize them [21, 22] or geometry images, and use a 2D CNN to analyze form using the frequently sampled feature images. Other methods [53] extend the CNN to graphs described by irregular triangle meshes. These methods are restricted to smooth manifold meshes, despite being robust to isometric deformation of 3D shapes. These methods are still computationally costly because of the local features they use. Bronstein et al. [53] provide a thorough overview of these strategies.

2.2.4 Point based methods

PointNet [2], PointNet++ [3], and [54] are examples of the later approach to adapting to 3D irregularity. PointNet was introduced by Su et al. [2] as the first neural network that absorbs 3D point clouds directly. PointNet is a relatively fast and robust system when it comes to rigid transformations and ordering of points. Its key flaw is that it relies solely on max-pooling for background information. To compensate for this flaw, Point Net++ was later created. [9] proposed So-Net, a permutation invariant architecture with orderless point clouds based on an unsupervised model. The development of a self-organizing map (SOM) to model the spatial distribution of point clouds is the central concept of So-Net. The input point cloud is represented by a single feature vector in the network. On each point and SOM node, the SOM is used to perform hierarchical feature extraction. [55] Introduce grouping techniques that identify point neighborhoods in the initial world space and the learned feature space to solve the problem of 3D semantic segmentation of unstructured point clouds using a deep learning architecture.

They use dedicated loss functions to help structure the learned point feature space by defining the neighborhood in an adaptive manner that is highly sensitive to local geometry by using *k*-means clustering on the input point cloud and then defining complex neighborhoods in the learned feature space using K-nearest neighbor (knn). PointSift, which is analogous to a SIFT, is proposed by [56]. The module attempts to encode knowledge about different orientations in a scale-adaptive manner. They obtain information from all points in the local neighborhood by integrating the pointSIFT module on the PointNet++ architecture, which demonstrate a high performance on segmentation task, rather than using K-nearest neighbor as used in PointNet++. Su et al. [11] Proposed SPLANet, a network structure that used an unordered point cloud and used a spatial convolution operator. Sparse bilateral convolutional layers with indexing structures are used in this approach to perform convolutions only on the sections of the lattice that have been occupied. The main advantage of SPLATnet is that, like regular CNN architectures, it allows for simple filter neighborhood specification.

2.3 Simplifying and sampling point clouds

There have been many techniques suggested in the literature for either point cloud simplification [57, 58] or sampling [59, 60]. Pauly et al. [57] introduced and evaluated multiple point-sampled surface simplification methods. Clustering processes, iterative simplification, and particle simulation are some of the techniques used. These algorithms generated a simplified point set that was not limited to being a subset of the original. To minimize the number of points, [59] proposed a view-dependent algorithm. In order to increase human understanding of the sampled point range, they used hidden-point elimination and target-point occlusion operators. Chen et al. [60] used graph-based filters to extract per-point features recently. A sampling strategy is likely to choose points that retain precise details. The sampling methods described above are designed to achieve a variety of sampling goals. They may not, however, take the task's goal into account explicitly.

3 Method

Because of its regularity, the volumetric grid is commonly used for 3D deep learning. Though, using lower-order local approximation functions like the piece-wise constant function to reflect finer geometry data, it requires a very high-resolution grid which may be inefficient in terms of memory and computation.

In this work, we propose a hybrid network that combines a point cloud and a voxel grid with a fixed number of points in each grid cell as a result, the network is able to learn higher-order local approximation functions that can better describe local geometry shape data.

We now present our hybrid network, beginning with a discussion of its sampling module (Sect. 3.1) and then moving on to its architecture for classification (Sect. 3.2) and segmentation (Sect. 3.3) tasks.

3.1 Sampling module

We use a point cloud-based occupancy grid, with the points that fall within each voxel grid serving as the voxel's key features. Unlike [14], which uses the occupancy grid as their primary form of 3D data representation. We use point clouds to build a voxel grid and then assign the points that fall into each voxel grid as the voxel's primary feature. Let k equals to number of points in each cell. However, each voxel can contain a different number of points.

To solve this problem, we use point quantization to ensure that each voxel has the same number of points. If the voxel contains more than K points, we sample K points from the total number of points within the voxel using the farthest points sampling technique. If the number of points in a voxel is less than K, we sample K points with substitution. As a result, the number of points in each voxel grid will be the same. This makes it simpler for 3D convolution kernels to extract object features because all voxels have the same feature size vector. Finally, we pad the voxel with zeros if the voxel has no point. Figure 1 shows the sampling steps of our method.

Given an input points within a voxel grid $a_1, a_2, ..., a_n$, we first select a subset of points $a_{i1}, a_{i2}, ..., a_{in}$ using FPS so that a_{ij} will be the most distant point from the set $a_{i1}, a_{i2}, ..., a_{ij-1}$ with regard to the remaining points. This method covers a larger number of points than random sampling.

Then, from the sampled points, we calculate the magnitude of the point (the Euclidean distance between points and their object's center, which is denoted by L), distance D, and angles between each embedded within each voxel to obtain the relationship between points within each voxel. For each voxel, the results are stacked as features, where D is the distance between each pair of points and θ is the sine of the angle between them. These three key variables define the characteristics of each voxel. As a result, a k-pointed cell will have (L, D, θ) k features. The computation of features can be expressed as follows:

$$L(K) = U_{i=1}^{k} |p_i|, |p_{i+1}| .... |p_k| \tag{1}$$

$$\theta = U_{i=1}^{k} sin^{-1}\left(\frac{p_i \cdot p_{i+1}}{|p_i| * |p_{i+1}|}\right), \\ sin^{-1}\left(\frac{p_i \cdot p_{i+2}}{|p_i| * |p_{i+2}|}\right), ...sin^{-1}\left(\frac{p_k \cdot p_1}{|p_k| * |p_1|}\right) \tag{2}$$

$$D(K) = U_{i=1}^{k} E(p_i, p_{i+1}), E(p_{i+1}, p_{i+2}) .... E(p_k, p1) \tag{3}$$

$$|p| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{4}$$

In Eq. 1, $|p|$ is the magnitude of point p and the magnitude of the point is computed using Eq. 4 where a_x, a_y and a_z are the corresponding value of x,y,z coordinate of point cloud in 3D space. $E(p_1, p_2)$ of Eq. 3 is Euclidean distance between two points (p_1, p_2) on 3D space. $(p_1.p_2)$ of Eq. 2 is the dot product between point p_1 and p_2.

3.2 Classification network

This network extracts global features from input voxel. It uses multiple convolutions with max. pooling to generate a variety of hierarchical features. A 5x5x5 kernel filter size and 18 convolutional filters. We use Relu [61] to help with output activation with batch normalization [62] that minimize the shift of internal-covariance. The pooling layers help to minimize overfitting and also drastically reduce the

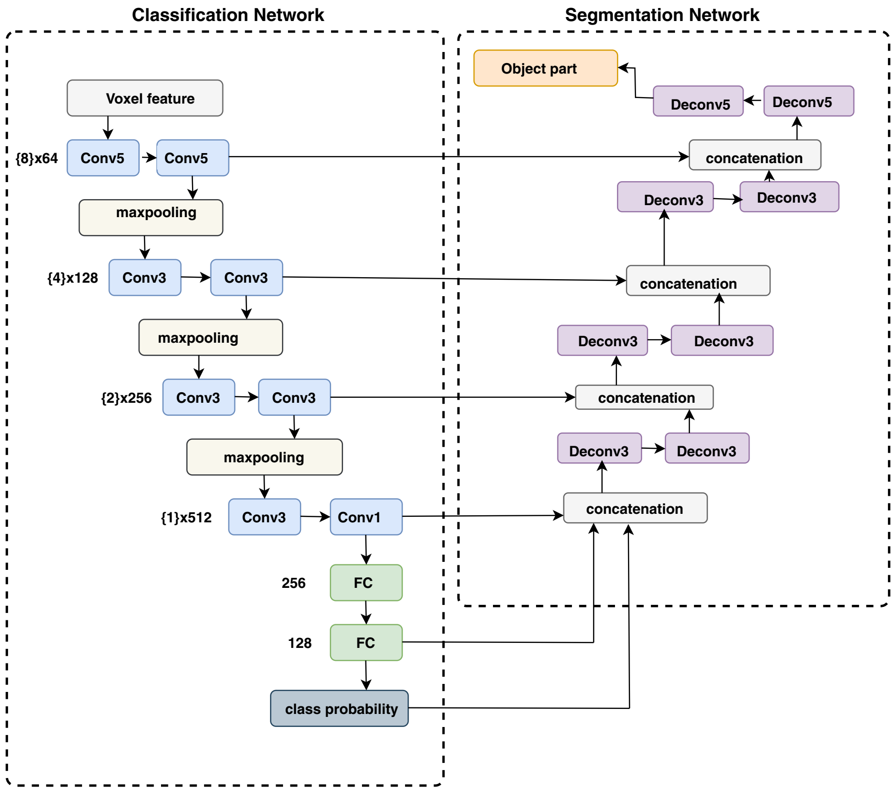

Fig. 2 Show both our model's classification and segmentation networks. The classification network extracts global features from the voxel feature's input. The network is made up of eight (8) convolution layers, a max-pooling operation after every two convolution layers, two (2) fully connected layers, and a fully connected layer that predicts the final class of the object. The segmentation network decodes the feature by up sampling and combining them to construct the object parts

computational cost. The final class of the object is predicted by the last fully connected layer. The notation conv5 means a convolutional layer with $5 \times 5 \times 5$ filter size, $\{16\} \times 18 = 16 \times 16 \times 16$ voxel size and 18 convolutional filters. Figure 2 shows the architecture of our network. Not only do the pooling layers have another form of translation invariance but also help to gradually shrink the representation's spatial scale in order to minimize the number of parameters and computational cost in the network and as a result, to limit overfitting. Our pooling layers are all max-pooling, which means they cut the grid size in each spatial dimension in half. Following a number of convolutional and max. pooling layers, our network's high-level reasoning is carried out in fully connected layers. In the end, an additional fully connected layer followed by a softmax is used to regress to each category's likelihood. This layer has the same number of nodes as the number of object groups in the dataset.

3.3 Segmentation network

The extracted features obtained from classification are decoded by this network to construct object parts. For effective global features, the network concatenates the high-level features from the object class likelihood and the last fully connected layers. The segmentation network is a mirror of the classification network, but with transposed convolutions instead of convolutions, and both networks are optimized at the same time. The classification network extracts and downsamples features, while the segmentation upsamples and fuses them together to generate the output. The classification network's initial features are combined with the segmentation network's equivalent decoded features to keep local sharp information within the same spatial resolution, and the network generates two labels for each voxel. For each cell, the segmentation network generates K + 1 labels. K labels correspond to the number of points in that cell, plus one additional cell-level label. To receive the cell-level ground truth labels for object parts, we select the label with the highest percentage of points in each cell. Cells that do not have a point are labeled as "no label." and all of the points within those cells are as well. If there are fewer than or equal to K points in each cell during testing, for each of them, we use the corresponding K labels. Otherwise, the cell-level label is applied to the remaining points.

4 Experiment

We applied our network to three tasks: classification, classification with noise and segmentation. We use the ModelNet [13] point cloud data given by [2] for classification and classification with noise. For segmentation, we used ShapeNet Part [63] and use the same training/testing split as [3]. For comparison with our suggested approach, random sampling is used as an alternative non-data based sampling method. Additional experimental details can be found in (Sect. 4.5)

4.1 Datasets

For the classification task, we use the ModelNet 10 and ModelNet 40 datasets [13], with accuracy as the evaluation metric and mean intersection over union (mIoU) on points is used to test ShapeNet Part [63].

- ModelNet[13]. Consist of two datasets of 3D CAD objects which are named ModelNet10 and ModelNet40. ModelNet40 have 9843 which are use for training while the remaining 2468 for testing with 40 classes in total while ModelNet10 have 3991 for training with 908 for testing with 10 classes in total.
- ShapeNetPart[63]. This dataset has 16 categories with 50 parts labeled and have 16,881 shapes in total.

4.2 Implementation details

The Tensorflow deep learning library was used to implement our proposed model in Python. All of our tests were run on a single NVIDIA Geforce GTX TITAN GPU with 3584 cores, CUDA 10.1 and cuDNN 7.1, as well as an Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz and 12GB RAM. The training took 36 hours for ModelNet10 and 72 hours for ModelNet40, respectively. For the ShapeNetPart segmentation, the training took 28 hours. We randomly rotate the object along the up-axis to augment the point cloud before sampling. We jitter the location of each point with a 0.02 standard deviation, zero mean, and Gaussian noise. A 32 batch size was used and a 0.5 initial decay of batch normalization with 0.99 batch normalization decay clipping. The weight for classification is 0.2 and 0.8 for segmentation.

$$L_{\text{cross}-entropy} = -\frac{1}{M}\left[\sum_{i=1}^{m}\sum_{j=1}^{n} 1\left\{y^{(i)} = j\right\} log\hat{y}^{(i)}\right] \tag{5}$$

$$L_{\text{total}} = \alpha L_{\text{classification}} + \beta L_{\text{segmentation}} \tag{6}$$

Equation 5 is a function with m as the total number of training data, n represents the total number of output neurons in the final layer. (e.g., n=10 if the total number of categories in the dataset is 10 and n=40 for the categories prediction of ModelNet40), while $y^{(i)}$ and $\hat{y}^{(i)}$ represents the true label and its corresponding prediction for *ith* output neuron, respectively. In Eq. 6, L_{total}, $L_{\text{classification}}$, and $L_{\text{segmentation}}$ represent the final loss of our model, which is a linear combination of both the loss of object classification prediction and the loss of object part segmentation, respectively (Fig. 5).

4.3 Classification on ModelNet10 dataset

For a fair comparison, we used [2] to preprocess the ModelNet10/40 datasets for our experiments. We used the default input points of 1024. Furthermore, we make an

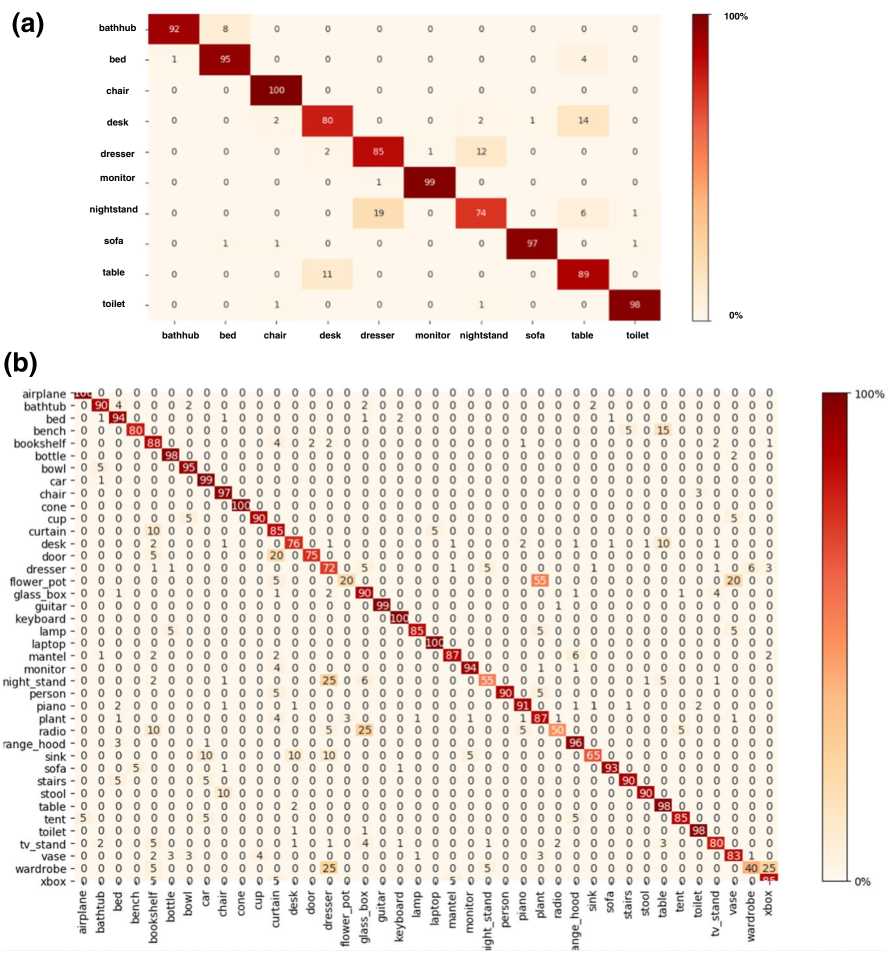

Fig. 3 Confusion matrix of our method **a** Confusion matrix of ModelNet10 **b** Confusion matrix of ModelNet40

effort to improve efficiency by using more points and surface normals as additional features.

Comparison The accuracy of state-of-the-art methods on ModelNet10 is shown in Table 2. Our network outperforms most other voxel-based approaches, including VoxNet [14], 3DShapeNet [13], 3DGAN [64] VSL [65], and binVoxNetPlus [66]. Although being inferior than the VRN-ensemble [15], which uses an ensemble of six models, each of which was trained independently over the course of six days on an NVidia Titan X. When compared to methods that use point clouds, our network outperforms G3DNet [67], PointNet [2], OctNet [20], and ECC [68]. Even though Point2Sequence outperforms ours and It uses the attention mechanism to learn the correlation of different areas in a local region, it does not propose a convolution on point clouds. One reason our method outperforms majority of the point cloud-based

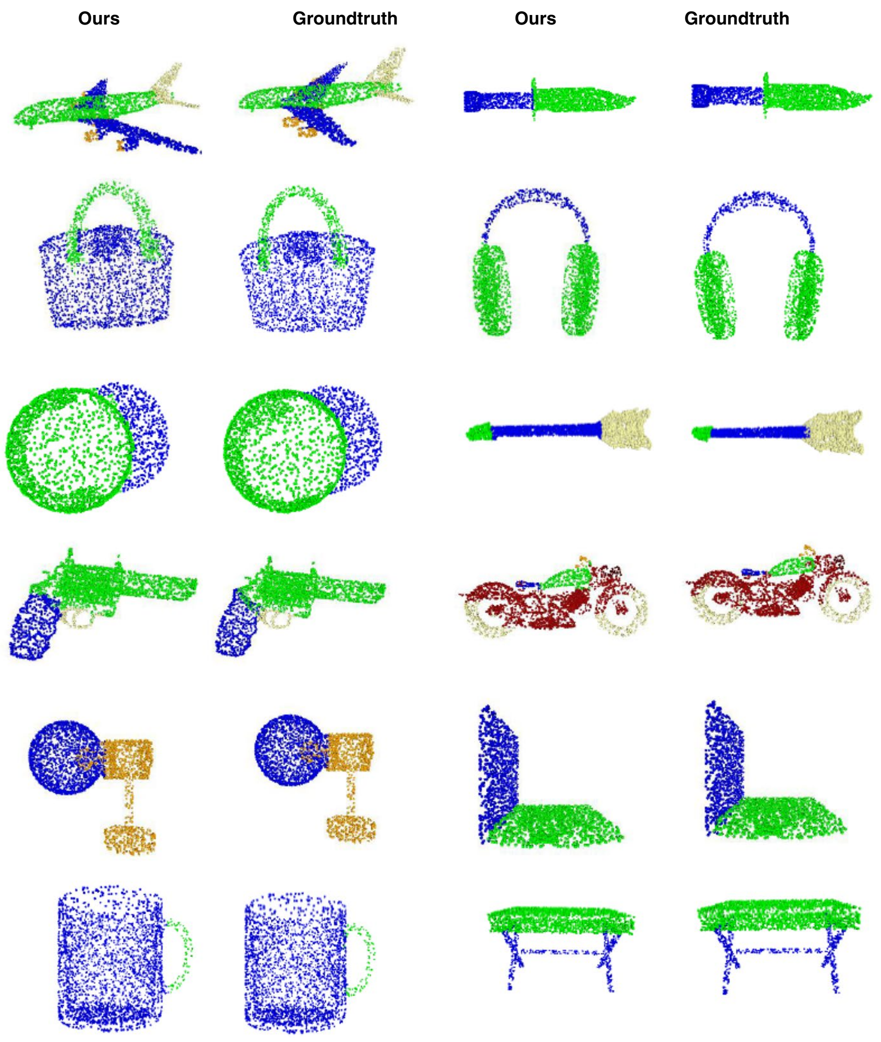

Fig. 4 Some visualize objects from our segmentation results on ShapeNet-part dataset First column: predicted segmentation, Second column: ground truth, third column predicted segmentation and fourth column : ground truth

methods is that it learns higher-level features by better capturing the contextual neighborhood of points. Regarding Multi view approaches, despite the fact that our approach outperforms DeepPano [24], OrthographicNet [69]. There is still a small gap between our method and the multi-view based methods SeqView2seqlabels [70], which could be due to the fact that these models can only perform well when the views are in a specific order rather than any kind of unordered views (Fig. 4).

4.4 Classification on ModelNet40 dataset

On the ModelNet40 dataset, our model obtains a compelling accuracy of 88.2%, as shown in Table 3. Our model is first compared to volumetric models. Table 3 shows that our model outperformed the majority of the volumetric models. 3DShapeNet [13] was the first model to explore the use of 3D volumetric voxels for 3D classification tasks on the ModelNet40 dataset. Our model outperforms this model by 11.2% in overall classification accuracy. In comparison to VoxNet [14], the model achieved 83%, which is less than our model accuracy with a margin of 5.2%. NormalNet [49] achieved an overall classification of 88.6% using two inputs to their model (normal vector and voxel grid), which is higher than our models with 0.4%. However, in ModelNet10, our model outperformed NormalNet by a margin of 0.3%. Our method outperforms LightNet [48] by 1.3%.

In comparison to multi-view-based network models, [24] achieved an accuracy of 82.5%, which is lower than our model by 5.7%. It is important to note that the model greatly benefits from the already advanced classical 2D CNN. Compared to our model, SeqView2seqlabels [70] achieved a classification accuracy of 93.4 percent, which is higher than ours. This may be because these models can only work well when the views are in a fixed order, rather than some other kind of unordered views. In comparison to point-based models, our model produced results that were comparable to the majority of the models presented. Point2sequence [71] had the highest classification accuracy of 92.6%, which is higher than our model's 4.4%. However, with a margin of 2.1 percent, our model outperforms DPRNet [72] and PointWise [12], while PointNet [2] and NPCEM [73] outperform our model with 1.0 percent and 1.2 percent, respectively. To further illustrate the effectiveness of our model, Fig. 3 shows the confusion matrix of our approach. The confusion matrix was normalized to 100%. We can clearly see that most objects from all classes are recognized correctly.

4.5 Analysis of using alternative sampling methods

To analyze the benefit of our sampling strategy, On ModelNet10 [13], we compare the classification accuracy of our method using various sampling/querying methods under various conditions. Using different voxel sizes, we change the number of points sampled (K) per voxel from 3 to 6. For center sampling method, we compare our approach to random point sampling (RPS). For neighbor querying method, we compare our method with K-nearest neighbors .In both cases, we use either k-nearest neighbor or RPS to replace our FPS. We also use the points that fall inside the voxel to calculate the centroid of the voxel without using any sampling method.

The points that are closest to the initial point are chosen for the k-nearest neighbor before we hit K sample. The aim is to sample points that are as close as possible to one another in a given voxel. For calculating the centroid of the voxel, we use the points that fall within the voxel and the centroid's coordinate as an input to our network. In this case, if a voxel only has one point, the point coordinate is used as the voxel's centroid. Otherwise, if there are more than one point, the centroid of the points is computed to obtain a single point coordinate (x; y; z) at their base. The centroid's point vector is fed into our deep network, which extracts global features. Table 4 summarizes the results of the qualitative and quantitative evaluations. In all cases, our FPS-based approach outperforms other methods in terms of classification accuracy (10 percent more than RPS). When K is very high, KNN has no advantage over FPS.

The following are some of the factors that favor our approach:

- Rather than sampling centers from N points, our method starts with a point in the set and iteratively selects the farthest point from the points already selected. This method has the advantage of covering the entire point set in a given number of centroids.
- Each occupied voxel contains the same number of points. The technique decreases the coverage loss caused by density imbalance in a local region since the points are more uniformly distributed.

4.6 Part segmentation on ShapeNetPart

We tested our model's performance on a 3D object component segmentation task to further validate its performance on 3D shape understanding. 3D semantic parts segmentation attempts to predict the correct labeling of object parts such as the tail, wing, and engine in the case of an airplane object. As an assessment metric, we used the mean Intersection over Union (mIoU) proposed in [2]. For each part shape in the object category, we measure the union between ground-truth and prediction for each shape. To calculate the mIoU for each object category, we compute the average of all object mIoUs in the object category. The average mIoUs of all test objects are also used to measure the overall mIoU. A cross-entropy loss is used to optimize the segmentation training process, just as it is for our model's 3D object classification task.

Comparison Our model achieves mean IoU of 83% using a voxel of size 16x16x16. Our model outperforms KD-Net [19] by 5.9% and 3D-CNN [75] by 3.3%, as shown in Table 5. With 0.6% and 1.9%, respectively, PointNet [2] and PointNet++ [3] outperform our model. Our model outperformed PointNet++ in four categories, while PointNet++ only outperformed our model in one (motorbike). In contrast to learning2segment [74], which achieved the highest mIoU on the ShapeNet-part dataset, Our model still outperforms this model in four categories: bag, cap, earphone, and mug. It's also worth noting that this model converts 3D point

Table 2 Classification accuracy on ModelNet10 dataset

Method	Representation	Parameter	Acc. (%)
3DshapeNet [13]	Volumetric	38M	83.5
VoxNet [14]	Volumetric	0.9M	92.0
NormalNet [49]	Volumetric	6.5M	93.1
3D-GAN [64]	Volumetric	–	91.0
VSL [65]	Volumetric	–	91.0
lightNet [48]	Volumetric	–	93.3
binVoxNetPlus [66]	Volumetric	–	92.3
VRN-ENSEMBLE [15]	Volumetric	–	**97.1**
OctNet [20]	Point cloud	–	90.4
PointNet [2]	Point cloud	–	77.6
Point2Sequence [71]	Point cloud	–	95.3
ECC [68]	Point cloud	–	90.0
G3DNet [67]	Point cloud	–	93.1
OrthographicNet [69]	Multi-view	–	88.5
SeqView2seqlabels [70]	Multi-view	–	94.8
DeepPano [24]	Multi-view	–	88.6
Ours	point-vol.	8M	93.4

The approach with the highest classification accuracy has been highlighted in bold

clouds to 2D matrices before applying classic 2D convolution. This method may not scale well on large scale lidar point clouds because projecting such data will result in too much noise, which may lead to object structural information loss. Figure 4 shows some segmentation results of our model from ShapeNetPart dataset. As we can see, in the majority of cases, our results are visually appealing. For example, our method can separate a motorcycle's wheels from its body. Other models, such as the aeroplane, pistol, bag and cup, can be observed in a similar manner.

4.7 Shape classification with noise

Majority of the methods achieved satisfactory performance on synthetic datasets without corrupting the models. We add a Gaussian noise $N(0, \sigma)$ ranging from [0.1,0.5] during testing to demonstrate the effectiveness of our method. Figure 5 shows the robustness to noise of our approach in both ModelNet10 and ModelNet40 datasets.

4.8 Computational time

Table 6 demonstrates our average testing time for classification and segmentation. As we can see, despite having the same voxel resolution of 32^3, our model with a voxel size of 32^3 is faster than 3D CNN methods and still outperforms it in terms

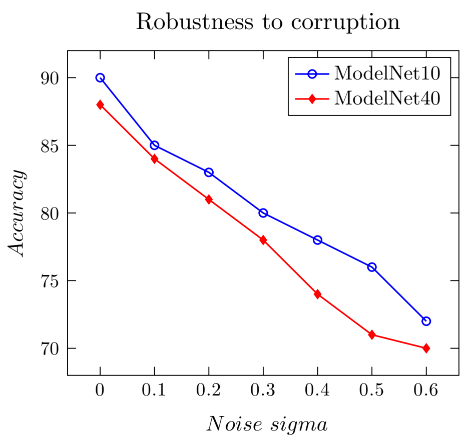

Fig. 5 During testing, we apply a Gaussian noise $N(0, \sigma)$ ranging from [0.1,0.5], demonstrating that our approach is robust to noise in both the ModelNet10 and ModelNet40 datasets[13]

of accuracy as shown in Table 5 Despite being slower than [2], our method outperforms it in classification tasks on ModelNet10, as shown in Table 2.

5 Conclusion

We demonstrated a method for optimizing point cloud and voxel data for a subsequent task. The method entails simplifying point clouds to create a voxel grid and then assigning the points that fall into each voxel grid as the primary feature of the voxel. We also design a sampling module that uses the magnitude of the point (the Euclidean distance between the point and the object's center) as well as the angles between each point embedded within each voxel to determine the relationship between points within each voxel. Experiments on standard benchmark datasets: ModelNet10, ModelNet40 [13] and ShapeNetPart[63] show that our method

Table 3 Classification accuracy on ModelNet40 dataset

Method	Representation	Parameter	Acc. (%)
3DshapeNet [13]	Volumetric	38M	77.0
VoxNet [14]	Volumetric	0.9M	83.0
NormalNet [49]	Volumetric	6.5M	88.6
lightNet [48]	Volumetric	–	86.9
PointNet [2]	Point cloud	–	89.2
DPRNet [72]	Point cloud	–	86.1
PointWise [12]	Point cloud	–	86.1
NPCEM [73]	Point cloud	–	89.4
Point2Sequence [71]	Point cloud	–	92.6
SeqView2seqlabels [70]	Multi-view	–	**93.4**
DeepPano [24]	Multi-view	–	82.5
Ours	point-vol.	8M	88.2

The approach with the highest classification accuracy has been highlighted in bold

Table 4 Performance comparisons of different sampling methods, run on ModelNet10[13]

Voxel size	FPS		KNN		Centroid		RPS	
	3	6	3	6	3	6	3	6
$8 \times 8 \times 8$	87.5	88.4	87.7	88.9	87.1	88.4	84.7	85.9
$16 \times 16 \times 16$	88.3	90.3	89.7	90.7	86.6	87.0	88.3	87.9
$32 \times 32 \times 32$	88.7	93.4	89.4	91.1	88.0	88.3	87.1	92.0

Table 5 Segmentation results of different methods on ShapeNet-part dataset

Methods	Ours	P.Net [2]	P.Net++ [2]	KD-Net[18]	L.2seg. [74]	3DCNN[75]	Point-Grid[76]	O-CNN[19]
mIoU	83.1	83.7	85.1	77.2	88.8	79.4	86.4	85.9
Airplane	73.9	83.4	82.4	79.9	86.5	75.1	85.7	85.5
Bag	88.5	78.7	79.0	71.2	78.9	72.8	82.5	87.1
Cap	91.6	82.5	87.7	80.9	83.4	73.3	81.8	84.7
Car	71.6	74.9	77.3	68.8	80.9	70.0	77.9	77.0
Chair	87.5	89.6	90.8	88.0	92.6	87.2	92.1	91.1
Earphone	85.7	73.0	71.8	72.4	77.6	63.5	82.4	85.1
Guitar	86.2	91.5	91.0	88.9	93.3	88.4	92.7	91.9
Knife	86.1	85.9	85.9	86.4	91.6	79.6	85.8	87.4
Lamp	89.1	80.8	83.7	79.8	89.0	74.4	84.2	83.3
Laptop	93.2	95.3	95.3	94.9	96.7	93.9	95.3	95.4
Motorbike	60.3	65.2	71.6	55.8	70.0	58.7	65.2	56.9
Mug	94.0	93.0	94.1	86.5	87.2	91.8	93.4	96.2
Pistol	79.6	81.2	81.3	79.3	84.5	76.4	81.7	81.6
Rocket	46.4	57.9	58.7	50.4	58.8	51.2	56.9	53.5
Skate-board	69.6	72.8	76.4	71.1	83.0	65.3	73.5	74.1
Table	83.3	80.6	82.6	80.2	88.1	77.1	84.6	84.4

Table 6 Average testing time of our method with others on ModelNet40 [3]

Method	Classification (ms)	Segmentation (ms)
PointNet [2]	9	28
PointNet++ [3]	163	–
PonitGrid (16^2)[76]	28	64
3DCNN[75] (64^2)	49	137
Ours (32^3)	28	19

favorably compares over some deep learning approaches VoxNet [14], 3DShapeNet [13], 3DGAN [64] VSL [65], G3DNet [67], DeepPano [24], OrthographicNet [69] and binVoxNetPlus [66]. Furthermore, our model can distinguish 3D objects while using substantially less memory. Because of its simple structure and small number of parameters, our model is ideal for real-time object classification.

Acknowledgements This work was supported by the National Natural Science Foundation of China (61671397). We thank all anonymous reviewers for their constructive comments.

References

1. Gezawa AS, Zhang Y, Wang Q, Yunqi L (2020) A review on deep learning approaches for 3d data representations in retrieval and classifications. IEEE Access 8:57566–57593
2. Qi Charles, Su H, Mo K, Guibas L (2017) PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, pp. 77–85
3. Qi Charles, Yi L, Su H, Guibas L (2017) PointNet++: Deep hierarchical feature learning on point sets in a metric space In: Advances in Neural Information Processing Systems, pp. 5100–5109
4. Li Y, Bu R, Sun M, Wu W, Di X, Chen B (2018) PointCNN: Convolution on X-transformed points. Proc Adv Neural Inf Process Syst (NIPS) 31:820–830
5. Manzil Z, Satwik K, Siamak R, Barnabás P, Ruslan S, Alexander JS (2017) Deep sets. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 3394–3404
6. Shen Y, Feng C, Yang Y, Tian D (2018) Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
7. Wang D, Posner I (2015) Voting for voting in online point cloud object detection. Robot: Sci Syst 1:10–15607
8. Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein M, Solomon J (2019) Dynamic graph CNN for learning on point clouds. ACM Trans Gr (TOG) 38:1–12
9. Li J, Chen B, Lee GH (2018) SO-Net: Self-organizing network for point cloud analysis. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:9397–9406
10. Tchapmi LP, Choy C, Armeni I, Gwak J, Savarese S (2017) SEGCloud: Semantic segmentation of 3D point clouds. In: 2017 International Conference on 3D Vision (3DV), pp. 537–547
11. Su H, Jampani V, Sun D, Maji S, Kalogerakis E, Yang M, Kautz J (2018) SPLATNet: Sparse lattice networks for point cloud processing. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:2530–2539
12. Hua B, Tran M, Yeung S (2018) Pointwise convolutional neural networks. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:984–993
13. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3D ShapeNets: A deep representation for volumetric shapes. IEEE Conf Computer Vision Pattern Recognit (CVPR) 2015:1912–1920
14. Maturana D, Scherer S (2015) VoxNet: A 3D convolutional neural network for real-time object recognition. IEEE/RSJ Int Conf Intell Robots Syst (IROS) 2015:922–928
15. Brock A, Lim T, Ritchie J, Weston N (2016) Generative and discriminative voxel modeling with convolutional neural networks.arXiv:1608.04236
16. Eldar Y, Lindenbaum M, Porat M, Zeevi Y (1997) The farthest point strategy for progressive image sampling. IEEE Trans Image Process: Publ IEEE Signal Process Soc 6(9):1305–15
17. Li Y, Pirk S, Su H, Qi C, Guibas L (2016) FPNN: Field probing neural networks for 3D Data.arXiv:1605.06240
18. Klokov R, Lempitsky V (2017) Escape from cells: deep Kd-networks for the recognition of 3D point cloud models. IEEE Int Conf Computer Vision (ICCV) 2017:863–872
19. Wang P-S, Liu Y, Guo Y-X, Sun C-Y, Tong X (2017) O-CNN: Octree-based convolutional neural networks for 3d shape analysis. ACM Trans Gr 36(4):1–11

20. Riegler G, Ulusoy AO, Geiger A (2017) OctNet: Learning deep 3D representations at high resolutions. IEEE Conf Computer Vision Pattern Recognit (CVPR) 2017:6620–6629
21. Masci J, Boscaini D, Bronstein M, Vandergheynst P (2015) Geodesic convolutional neural networks on riemannian manifolds. IEEE Int Conf Computer Vision Workshop (ICCVW) 2015:832–840
22. Boscaini D, Masci J, Rodolá E, Bronstein M (2016) Learning shape correspondence with anisotropic convolutional neural networks. In: NIPS
23. Bai S, Bai X, Zhou Z, Zhang Z, Latecki L (2016) GIFT: A real-time and scalable 3D shape search engine. IEEE Conf Computer Vision Pattern Recognit (CVPR) 2016:5023–5032
24. Shi B, Bai S, Zhou Z, Bai X (2015) DeepPano: Deep panoramic representation for 3-D shape recognition. IEEE Signal Process Lett 22:2339–2343
25. Su H, Maji S, Kalogerakis E, Learned-Miller E (2015) Multi-view convolutional neural networks for 3D shape recognition. IEEE Int Conf Computer Vision (ICCV) 2015:945–953
26. Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva CT (2001). Point set surfaces. In: Proceedings of the conference on Visualization '01 (VIS '01). IEEE Computer Society, USA, 21–28
27. Lars L (2001) Point cloud representation, Technical Report, Faculty of Computer Science, University of Karlsruhe
28. Guo K, Zou D, Chen X (2015) 3D Mesh labeling via deep convolutional neural networks. ACM Trans Gr (TOG) 35:1–12
29. Sinha A, Bai J, Ramani K (2016) Deep learning 3D shape surfaces using geometry images. In: ECCV
30. Steinke F, Schölkopf B, Blanz V (2006) Learning dense 3D correspondence. In: NIPS
31. Sun J, Ovsjanikov M, Guibas L (2009) A concise and provably informative multi-scale signature based on heat diffusion. Computer Gr Forum 28:1383–1392
32. Rustamov R (2007) Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In: Symposium on Geometry Processing
33. Ovsjanikov M, Bronstein A, Bronstein M, Guibas L (2009) Shape google: a computer vision approach to isometry invariant shape retrieval. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 320–327
34. Golovinskiy A, Kim VG, Funkhouser T (2009) Shape-based recognition of 3D point clouds in urban environments. In: 2009 IEEE 12th International Conference on Computer Vision, 2154–2161
35. Wu Z, Shou R, Wang Y, Liu X (2014) Interactive shape co-segmentation via label propagation. Comput Gr 38:248–254
36. Yamauchi H, Lee S, Lee Y, Ohtake Y, Belyaev A, Seidel H (2005) Feature sensitive mesh segmentation with mean shift. In: International Conference on Shape Modeling and Applications 2005 (SMI' 05), 236–243
37. Vieira M, Shimada K (2005) Surface mesh segmentation and smooth surface extraction through region growing. Comput Aided Geom Des 22:771–792
38. Kazmi IK, You L, Zhang J (2013) A survey of 2D and 3D shape descriptors. In: 2013 10th International Conference Computer Graphics, Imaging and Visualization, 1–10
39. Rostami R, Bashiri FS, Rostami B, Yu Z (2019) A survey on data-driven 3D shape descriptors. Computer Gr Forum 38:356–393
40. Toldo R, Castellani U, Fusiello A (2009) Visual vocabulary signature for 3D object retrieval and partial matching. In: 3DOR@Eurographics
41. Nair V, Hinton GE (2009) 3D Object recognition with deep belief nets. NIPS 22:1339–1347
42. Alain G, Bengio Y (2014) What regularized auto-encoders learn from the data-generating distribution. J Mach Learn Res 15:3563–3593
43. Socher R, Huval B, Bath BP, Manning CD, Ng A (2012) Convolutional-recursive deep learning for 3D object classification. NIPS 25:656–664
44. Graham B (2015) Sparse 3D convolutional neural networks. BMVC
45. Riegler G, Ulusoy AO, Bischof H, Geiger A (2017) OctNetFusion: Learning depth fusion from data. In: 2017 International Conference on 3D Vision (3DV),pp. 57–66
46. Wang P, Liu Y, Tong X (2020) Deep octree-based CNNs with output-guided skip connections for 3D shape and scene completion. IEEE/CVF Conf Computer Vision Pattern Recognit Workshops (CVPRW) 2020:1074–1081
47. Bribiesca E (2008) A method for representing 3D tree objects using chain coding. J Vis Commun Image Represent 19:184–198
48. Zhi S, Liu Y, Li X, Guo Y (2018) Toward real-time 3D object recognition: a lightweight volumetric CNN framework using multitask learning. Comput Graph 71:199–207

49. Wang C, Cheng M, Sohel F, Bennamoun M, Li J (2019) NormalNet: A voxel-based CNN for 3D object classification and retrieval. Neurocomputing 323:139–147
50. Han Z, Shang M, Liu Y, Zwicker M (2019) View inter-prediction GAN: unsupervised representation learning for 3D shapes by learning global shape memories to support local view predictions. In: AAAI
51. Kanezaki A, Matsushita Y, Nishida Y (2018) RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:5010–5019
52. Feng Y, Zhang Z, Zhao X, Ji R, Gao Y (2018) GVCNN: Group-view convolutional neural networks for 3D shape recognition. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:264–272
53. Bronstein M, Bruna J, LeCun Y, Szlam AD, Vandergheynst P (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Process Mag 34:18–42
54. Yi L, Su H, Guo X, Guibas L (2017) SyncSpecCNN: Synchronized Spectral CNN for 3D shape segmentation. IEEE Conf Computer Vision Pattern Recognit (CVPR) 2017:6584–6592
55. Engelmann F, Kontogianni T, Schult J, Leibe B (2018) Know what your neighbors Do: 3D semantic segmentation of point clouds.arXiv:1810.01151
56. Jiang M, Wu Y, Lu C (2018) PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation.arXiv:1807.00652
57. Pauly M, Gross M, Kobbelt L (2002) Efficient simplification of point-sampled surfaces. IEEE Visualization 2002. VIS 2002:163–170
58. Moenning C, Dodgson N (2003) A new point cloud simplification algorithm
59. Katz S, Tal A (2013) Improving the visual comprehension of point sets. IEEE Conf Computer Vision Pattern Recognit 2013:121–128
60. Chen S, Tian D, Feng C, Vetro A, Kovacevic J (2018) Fast resampling of three-dimensional point clouds via graphs. IEEE Trans Signal Process 66:666–681
61. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: ICML
62. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift.arXiv:1502.03167
63. Yi L, Kim VG, Ceylan D, Shen I, Yan M, Su H, Lu C, Huang Q, Sheffer A, Guibas L (2016) A scalable active framework for region annotation in 3D shape collections. ACM Trans Gr (TOG) 35:1–12
64. Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J (2016) Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: NIPS
65. Liu S, Giles CL, Ororbia A (2018) Learning a hierarchical latent-variable model of 3D shapes. In: 2018 International Conference on 3D Vision (3DV), pp. 542–551
66. Ma C, An W, Lei Y, Guo Y (2017) BV-CNNs: Binary volumetric convolutional networks for 3D object recognition. BMVC 1:4
67. Dominguez M, Dhamdhere R, Petkar A, Jain S, Sah S, Ptucha R (2018) General-purpose deep Point cloud feature extractor. In: IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, pp. 1972–1981, https://doi.org/10.1109/WACV.2018.00218.
68. Simonovsky M, Komodakis N (2017) Dynamic edge-conditioned filters in convolutional neural networks on graphs. IEEE Conf Computer Vision Pattern Recognit (CVPR) 2017:29–38
69. Kasaei H (2019) OrthographicNet: A deep learning approach for 3D object recognition in open-ended domains.arXiv:1902.03057
70. Han Z, Shang M, Liu Z, Vong C, Liu Y, Zwicker M, Han J, Chen C (2019) SeqViews2SeqLabels: Learning 3D global features via aggregating sequential views by RNN with attention. IEEE Trans Image Process 28:658–672
71. Liu X, Han Z, Liu Y, Zwicker M (2019) Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network. In: AAAI
72. Arshad S, Shahzad M, Riaz Q, Fraz M (2019) DPRNet: Deep 3D point based residual network for semantic segmentation and classification of 3D point clouds. IEEE Access 7:68892–68904
73. Song Y, Gao L, Li X, Shen W (2020) A novel point cloud encoding method based on local information for 3D classification and segmentation. Sensors (Basel, Switzerland) 20:2501
74. Lyu Y, Huang X, Zhang Z (2020) Learning to segment 3D point clouds in 2D image space. IEEE/CVF Conf Computer Vision Pattern Recognit (CVPR) 2020:12252–12261
75. Leng B, Liu Y, Yu K, Zhang X, Xiong Z (2016) 3D object understanding with 3D convolutional neural networks. Inf Sci 366:188–201
76. Le T, Duan Y (2018) PointGrid: A deep network for 3D shape understanding. IEEE/CVF Conf Computer Vision Pattern Recognit 2018:9204–9214

Authors and Affiliations

Abubakar Sulaiman Gezawa[1] · **Zikirillahi A. Bello**[1] · **Qicong Wang**[1,2] · **Lei Yunqi**[1]

Abubakar Sulaiman Gezawa
gezawaabubakar@stu.xmu.edu.cn

1 Department of Computer Science, School of Informatics, Xiamen University, Xiamen 361005, China

2 Shenzhen Research Institute, Xiamen University, Shenzhen 518000, China