IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 July 2022, accepted 1 August 2022, date of publication 8 September 2022, date of current version 3 March 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3205107

RESEARCH ARTICLE

PCAPN: An Enhanced Feature Extraction Framework for Point Cloud

YULIN JI, JIANDAN ZHONG, YINGXIANG LI, JUNJIE FU, AND JIAWEI LIU
School of Communication Engineering, Chengdu University of Information Technology, Chengdu 610200, China
Corresponding author: Jiandan Zhong (heartbeat_ji@outlook.com)

ABSTRACT Point cloud is a widely used geometric data structure in the missions of 3D reconstruction, digital city and geologic survey etc. Extracting sufficient information from the point cloud is the key to deal with the aforementioned missions. However, huge number of points lead to computational complexity and inefficiency during the training process. To deal with this problem, this paper proposes a novel framework name Principal Component Analysis Point Net (PCAPN) for the feature extraction of point cloud. Firstly, a sampling module namely Component Point Sampling (CPS) is designed for generating several candidate sets of points by different scales, which defines the centroids of local regions. Secondly, a feature extraction framework based on MLP structure is adopted for extracting the feature vectors from the points set generated from the sampling module. Finally, the extracted feature vectors are concatenated together, and then put into a fully connected layer for classification. The proposed framework was evaluated on 2 benchmarks, i.e. ShapeNet part data set and ModelNet40. The experimental results show that our framework is efficient and robust. In particular, the results are significantly better than those obtained by the state-of-the-art frameworks. Our network is 4.6% more accurate than PointNet and 1.1% higher than PointNet++.

INDEX TERMS Principal component analysis point net, component point sampling, shapeNet part data set, feature extraction network, modelNet40.

I. INTRODUCTION

In recent years, with the maturity of simultaneous localization and mapping (SLAM) technology, Kinect technology and laser scanning technology, the amount of 3D point cloud data is also growing rapidly. How to describe the high-level semantic understanding of 3D point cloud data has attracted increasing attention. Semantic segmentation of 3D point cloud is a typical computer vision [1], [2], [3] problem, which refers to taking some original 3D point cloud data as input and converting it into a mask with highlighted regions of interest through a series of technical operations. It has become a research hotspot in the fields of automatic driving [4], navigation and positioning, and smart city [5], [6], [7], [8] etc.

There is rare research on the deep learning on point sets. PointNet [9] is a pioneering effort that directly processes point sets. The basic process of PointNet is to learn a spatial encoding of each point and then aggregate all individual point features to a global point cloud signature. So, PointNet does not capture local structure induced by the metric. However, exploiting local structure has proven to be important for the success of convolutional architectures. Because it has some drawbacks in some places, another method introduces a hierarchical neural network, named as PointNet++ [10], to process a set of points sampled in a metric space in a hierarchical fashion. The basic principle of PointNet++ is simple to understand. Firstly, the set of points is partitioned into overlapping local regions by the distance metric of the underlying space. Similar to convolutional neural network (CNN) [11], [12], local features capturing fine geometric structures are extracted from small neighborhoods. Such local features are further grouped into larger units and processed to produce higher level features. This process is repeated until we obtain the features of the whole point set.

Although PointNet++ solves many problems and recent articles have proposed many algorithms (i.e. PointCNN, PointWeb, PointNGCNN, PointConv, RandLA-Net) [13], [14], [15], [16], [17] for semantic segmentation of point

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

clouds, no discussion has been made on farthest point sampling(FPS). Although FPS has good point coverage, the selection of the FPS point has some randomness. Selecting particles only through Euclidean distance has a high probability of extracting the characteristics of similar points, resulting in the computational redundancy. Secondly, the feature extraction network used by PointNet++ is single-scale, which limits the comprehensiveness of feature extraction and makes the final accuracy to be improved. For the problems existing in PointNet++, we put forward many solutions, and we put forward our solutions.

In this research, we present a principal component analysis neural network called PCAPN. As shown in Fig. 3, our PCAPN framework is divided into five major layers, i.e. Sampling layer, Grouping layer, Point Net layer, Splicing layer and Fully-Connected layer to process multisets of points sampled in measurement space in a multi-scale manner. The general idea of PCAPN is simple: First, all the points in the point set are sorted according to the contribution ratio of the points in the measurement space, to obtain an ordered point set. Here, ordering is by the contribution ratio of the points to the point cloud. Then, multiple sets of points are selected from this set of ordered points as our particles (which are not duplicated to avoid redundancy in computation). Next, the characteristic information of each set of points is extracted by putting them into several groups. Finally, the feature information from each set of particles is stitched together to obtain the feature information of the entire point cloud. Compared with other state-of-the-art frameworks, the proposed multi-scale network can extract more features, avoid duplicate extraction in the process of extracting features, reduce computational redundancy. Our network is more efficient, feature extraction is more comprehensive, and improve the training efficiency is improved.

II. RELATED WORK

3D data has a variety of popular representations (e.g.: multi-views, point clouds, voxels), which has led to a variety of learning methods. Volume CNNs [18], [19], [20]: a pioneer in applying three-dimensional CNNs to voxelized shapes. However, volume representation is limited by its resolution due to data sparsity and computing costs. Fuzzy Polynomial Neural Network (FPNN) [21] and Vote3D [22] use special methods to deal with the sparsity problem. However, they still operate on sparse volumes, so dealing with very large point clouds can be a challenge. Multi-view CNNs: try rendering a three-dimensional point cloud or shape as a two-dimensional image, then classify it using a two-dimensional convolution network. These well-designed CNNs have achieved dominant performance in shape classification and retrieval tasks. However, these methods are currently limited to manifold meshes (such as organic objects), and how to extend them to non-isometric shapes (such as furniture) is not obvious. Feature-based DNNs [23], [24], [25] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully-connected net to classify the shape. We believe that they are limited by the representation ability of extracted features.

Recent publications [26], [27] have examined how in-depth learning can be applied to disordered sets. Even if point sets do have a basic distance metric, they will ignore the basic distance metric. Therefore, they cannot capture the local context of points, and are very sensitive to global set transformation and normalization. In this work, we solve these problems by explicitly considering the potential amount of information of each point in the design for the points sampled from the metric space. Since points sampled from measurement space are usually noisy and have uneven sampling density, this will affect effective feature extraction and cause learning difficulties. One of the key problems in point feature extraction is to select an appropriate scale. Prior to this, several methods have been developed in the world of geometry processing or photogrammetry and remote sensing. Compared with all this work, our approach learns to extract point features and balance multiple feature scales in an end-to-end manner. In three-dimensional measurement space, in addition to point sets, there are several commonly used in-depth learning representations, including volume grids [28], [29], [30] and geometries [31], [32], [33]. However, in all these efforts, the computation time and complexity are not explicitly considered.

Among all learning models, CNN is the most prominent one. However, convolution is not suitable for unordered point sets with distance measurement. Although PointNet and PointNet++ have partly solved this problem, due to the huge amount of point cloud data and its very important set structure, it is inevitable to select similar points as particles in the selection of particles. However, this will lead to the extraction of similar or same feature information through CNN learning, resulting in computational redundancy and reduced training accuracy. In order to solve this problem, we propose a new sampling method, namely CPS. The overall idea of CPS is to calculate the contribution rate of each point to the point cloud, and then get the importance of each point in the point cloud. According to this contribution rate, the original point clouds are sorted according to the order of contribution rate, so as to transform an unordered point cloud into an ordered one. Then, a group of points that best represent the original input point cloud are selected from this ordered set of points as our sampling points. In this way, we can avoid extracting similar or same points as our particles due to the huge amount of point cloud data. Here, we propose two forms of sampling points according to this idea. One is sampling points at equal intervals, which we call average point sampling (APS); the other is to extract some points with high contribution rate, which we call main point sampling (MPS).

PointNet++ proves the superiority of hierarchical learning. However, the idea of hierarchical learning adopted by PointNet++ is single-scale. This means, there is a probability that similar points will be selected as particles when selecting particles, resulting in insufficient information in

feature extraction. From this point of view, we construct the multi-scale hierarchical learning idea. Specifically, a number of different sets of points are sampled as particles at each scale, and the feature information extracted at each scale is joined into one feature information through hierarchical learning ideas. This allows us to extract to a large extent the feature information of the original input point cloud. The more feature information extracted, the better the training effect will be.

III. RESOLVENT

Our PCAPN must solve two problems: (a) How to ensure that the selected centroid of each scale is not repeated; (b) How to extract the feature information of point cloud as much as possible. Our work can be seen as the extension of PointNet++, which increases the multi-scale training structure, and also designs two different sampling methods based on one theory to make our PCAPN more effective. Firstly, mathematical knowledge is used to derive our sampling algorithms APS and MPS (section A). Then, we introduce the important structure of PointNet++ with deep-seated feature learning (Section B), which can learn features robustly and reduce the computational complexity to a certain extent.

A. MPS AND APS

First, we assume that the original point cloud is an n×m dimensional matrix. For example, formula (1).

$$X = \begin{pmatrix} \mathbf{x11} & \cdots & \mathbf{x1m} \\ \vdots & \ddots & \vdots \\ \mathbf{xn1} & \cdots & \mathbf{xnm} \end{pmatrix} \mathbf{n} \times \mathbf{m}. \tag{1}$$

In order to facilitate our subsequent calculation, a zero-mean is first applied, as shown in formula (2).

$$X = \begin{pmatrix} \mathbf{x_{11}} - \mathbf{u_1} & \cdots & \mathbf{x_{1m}} - \mathbf{u_m} \\ \vdots & \ddots & \vdots \\ \mathbf{x_{n1}} - \mathbf{u_n} & \cdots & \mathbf{x_{nm}} - \mathbf{u_m} \end{pmatrix} \mathbf{n} \times \mathbf{m}, \tag{2}$$

where $\mathbf{u_j} = \frac{1}{n}\sum_{j=1}^{\mathbf{n}} \boldsymbol{x_{ji}}$ represents the mean of the column vector; Our goal is to find the relationship between each point and other points from the original input point cloud, and then filter out similar points when selecting particles, which can avoid extracting duplicate information and computing redundancy. Now that we are looking for relationships between each point and other points, we first think about the covariance of the matrix. As shown in formula 3, the covariance of the original input point cloud is calculated.

$$\boldsymbol{C} = \frac{1}{\boldsymbol{n}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} = \begin{pmatrix} \boldsymbol{Cov}(\boldsymbol{x_1}, \boldsymbol{x_1}) & \cdots & \boldsymbol{Cov}(\boldsymbol{x_1}, \boldsymbol{x_k}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{Cov}(\boldsymbol{x_k}, \boldsymbol{x_1}) & \cdots & \boldsymbol{Cov}(\boldsymbol{x_k}, \boldsymbol{x_k}) \end{pmatrix}, \tag{3}$$

From the formula (4), it is clear that the two elements on the diagonal line of this matrix are the variance of two vectors, while the other elements are the covariance. They are unified into a matrix.

Based on the above derivation, we find that to achieve the desired effect, we are equivalent to diagonalizing the covariance matrix: that is, to make all the elements except the diagonal zero, and to arrange the elements on the diagonal from top to bottom in size, so that we can achieve the optimization goal. This may not be clear yet. Let's take a closer look at the relationship between the original matrix and the matrix covariance matrix after the base transformation: set the covariance matrix corresponding to the original data matrix **X** to **C**, and **P** to a set of rows of matrices, set **Y**=**PX**, then **Y** to **X** to base the transformed data of **P**. Setting the covariance matrix of **Y** to**D**, we deduce the relationship between **D** and **C**:

$$\boldsymbol{D} = \frac{1}{\boldsymbol{n}}\boldsymbol{YY}^{\boldsymbol{T}} = \frac{1}{\boldsymbol{n}}(\boldsymbol{PX})(\boldsymbol{PX})^{\boldsymbol{T}} = \boldsymbol{P}\left(\frac{1}{\boldsymbol{n}}\boldsymbol{XX}^{\boldsymbol{T}}\right)\boldsymbol{P}^{\boldsymbol{T}} = \boldsymbol{PCP}^{\boldsymbol{T}} \tag{4}$$

It is easy to see from formula (4) that the **P** to be found is not something else, but a **P** that diagonalizes the original covariance matrix. In other words, our goal is to find a matrix**P** that satisfies that $\mathbf{PCP^T}$ is a diagonal matrix and the diagonal elements are arranged from large to small, so the first **K** rows of **P** are the basis to be searched. Multiplying **X** by the matrix composed of the first **K** rows of **P** reduces **X** from **N** to **K** dimensions and satisfies the above optimization conditions.

So far, we have focused all our attention on the diagonalization of the covariance matrix, since covariance matrix **C** is a symmetric matrix, according to the properties of the symmetric matrix:

1) The eigenvectors corresponding to different eigenvalues of a real symmetric matrix must be orthogonal;
2) Set up eigenvectors λ If the multiplicity is **r**, there must be**r** linearly independent, eigenvectors corresponding to λ, Therefore, the **R** eigenvector units can be orthogonalized.

From the properties of the two real symmetric matrices above, we can see that a real symmetric matrix with n rows and **N** columns must find n unit orthogonal eigenvectors. If this **n** eigenvectors are set, we will make up the matrix with **E** = (**e1**, **e2**, …, **en**) columns. Then covariance matrix **C** concludes as follows:

$$\mathbf{E^T CE} = \mathbf{\Lambda} = \begin{pmatrix} \boldsymbol{\gamma_1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\gamma_n} \end{pmatrix}, \tag{5}$$

where **Λ** is a diagonal matrix and its diagonal elements are the eigenvalues corresponding to each eigenvector (which may be duplicated).

Here we find that we have found the required matrix P: **P**=**ETP**=**ET**. P is a matrix arranged by rows after the eigenvectors of the covariance matrix are united, where each row is a eigenvector of **C**. Here we have the eigenvalues of the covariance matrix. Let's recall our goal of giving each point in the original input point cloud a specific order, where we calculate the information contribution B_J as we think. As shown

in formula 6. Although there are equal eigenvalues in the covariance matrix, we can assume that it contributes equally to the information of our entire point cloud and that only one value can be retained when we sample. The remaining point clouds, as required by our sample points, are extracted from the remaining points. Here we propose two solutions: one is APS; The other is MPS. APS refers to the equal-difference extraction of a set of points from an ordered set of points. As shown in Figure 1, a set of points is extracted from an ordered set of points at equal intervals according to the number of samples we need. MPS refers to the point where we extract the largest contribution rate of information from this set of ordered points. As shown in Figure 2, according to the number of sample points we need, we extract the points with the largest contribution rate from the largest to the smallest.

$$\textit{Information contribution rate } b_j = \frac{\gamma_j}{\sum_{m=1}^{k} \gamma_m} \tag{6}$$

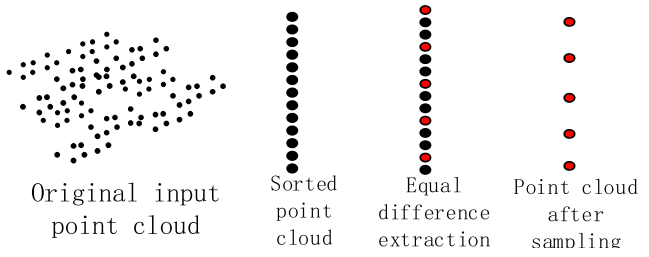


FIGURE 1. APS (The original input point cloud is given a sequence by the contribution rate of information, and then some points are extracted at medium intervals from the given sequence as the points after sampling. The middle black dot represents the point after sorting, and the right side represents equal interval to extract part of the point (red dot represents)).

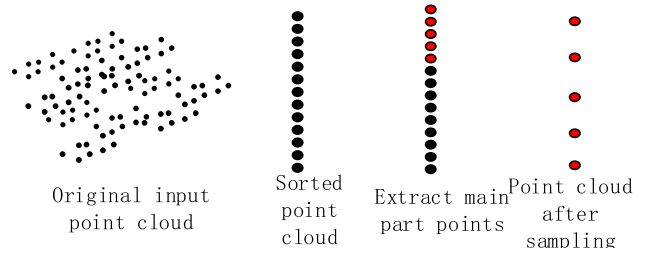


FIGURE 2. MPS (The original input point cloud is given a sequence by the contribution rate of information, and then a portion of the points with a higher contribution rate of information is extracted from the given order as the points after sampling. The middle black dot represents the point after sorting, and the right part represents the extracted point (red dot represents)).

B. FEATURE EXTRACTION NETWORK

PointNet++ builds a hierarchical feature extraction network, which consists of three key layers i.e. sampling layer, grouping layer and PointNet layer. The sampling layer selects a set of points from the input points to define the centroid of the local area. Then, the grouping layer constructs a local region set by finding ''adjacent'' points around the centroid. The PointNet layer encodes local regions as eigenvectors. This hierarchical network can extract features and the training results are good. However, since the idea is to filter out as many points as possible with similar information contributions and reduce computational redundancy, it is clear that the FPS used by PointNet++ does not meet our requirements. Moreover, a calculation is needed to select the centroid at each sampling time, which increases the computational redundancy and obviously does not meet the requirements of our design. For this reason, we have optimized the PointNet++ hierarchical network feature extraction, which be introduced in the following section.

Our multi-scale feature extraction network uses four key layers i.e. sampling layer, grouping layer, PointNet layer, and stitching layer. As shown in Fig. 3, the sampling layer uses APS and MPS sampling methods to sort and sample the points in the original input point cloud as the centroid of the local area. Then the grouping layer constructs a set of points in the local area by finding ''adjacent'' points around the centroid. The PointNet layer then codes the local area as a feature vector. Finally, the eigenvectors from each scale are stitched. The following part A will describe the specific role of each key layer separately.

1) SAMPLING LAYER

Given the input points $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, we need to extract features at multiple scales, So, we use MPS and APS to select a subset of point **{xi}**, that is, to find more than one set of the most representative particles from this point set in a given order as our particles. The main steps are as follows. Assuming that the original set of input points is sorted to obtain N points, we need m scales; The point m_1 required by the first set of scales is sampled from the ordered points as the particles in our first set of scales, and there are $(N\text{-}m_1)$ remaining points. Then the second set of points M2 is sampled from this$(N\text{-}m_1)$ point as the particles in our second set of scales. On the steps continues all our scale midpoints are sampled.

2) GROUPING LAYER

The input to this layer is a set of points of size $N \times (d + C)$ and a set of coordinates of the centroid of size $N_0 \times d$. The output is a set of points of size $N' \times K \times (d + C)$, where each set corresponds to a local region and K is the number of points in the neighborhood of the centroid point. It should be noted that K varies among different groups, but subsequent point network layers can convert a flexible number of points into a fixed-length local area eigenvector. In the point set sampled from the metric space, the neighborhood of the point is defined by the metric distance. Just like the grouping layer of PointNet++.

3) POINTNET LAYER

The PointNet layer here still uses the PointNet layer in PointNet++, where the input is $N' \times K \times (d + C)$ local area of the point. Each local area in the output is extracted from its centroid, local features encode its neighborhood, and the

output data size is $N' \times (d + C')$. The goal is to extract the characteristics of the largest pool for subsequent connections to the full connection layer.

4) STITCHING LAYER

We have sampled m scales together. It is obvious that we cannot calculate the score for each scale. Then we need to stitch the features from each scale to get our final features. The final set of features to the fully-connected layer to complete our classification scoring and our classification tasks.

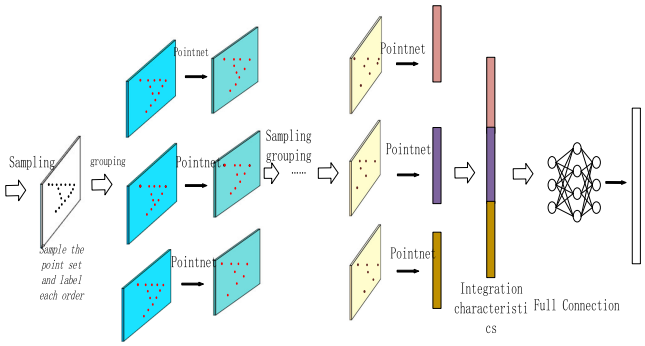


FIGURE 3. PCAPN network model (First, we will sample the point set, get the points after sampling, take our specific training data to say, each training set and dataset of ModelNet40 are 10,000 points, where we sort the 10,000 points once (note that only one sort can be done here), and then select 1024 points, 512 points and 256 points for these sorted points respectively; These points are then grouped and PointNet operated, and the intermediate sampling, grouping, and PointNet layers are divided three times in our actual training. Finally, the information obtained from these three sampling points is stitched and fed into the full-connected layer for classification.)

IV. EXPERIMENT

The proposed network was evaluated in four ways. First, the MIou evaluation method was used to evaluate the model on the ShapeNet part dataset (Sec A). Then the PointNet++ model was adopted to compare our sampling algorithm with the FPS (Sec B). Nest, detailed experiments were conducted to validate our network design (Sec C). Finally, we show some visualization of content (Sec D).

ModelNet40: 40 categories of CAD models (mainly man-made), 9843 shapes for training, and 2468 shapes for testing datasets as our main training and testing dataset for this experiment.

ShapeNet Part Data Set: contains 16,881 shapes from 16 categories, annotated with 50 parts in total. Most object categories are labeled with two to five parts. Ground truth (GT) annotations are labeled on sampled points on the shapes.

A. mIoU EVALUATION METHOD

Part segmentation is a challenging fine-grained 3D recognition task. Given a 3D scan or a mesh model, the task is to assign part category label (e.g. chair leg, cup handle) to each point or face. The proposed network was evaluated on ShapeNet part data set from [34]. We formulate part segmentation as a per-point classification problem. Evaluation metric is mIoU on points. For each shape S of category C, the shape's mIoU is calculated as follows: For each part type in category C, IoU between GT and prediction compute. If the union of GT and prediction points is empty, then part IoU is counted as 1. Then, the IoUs for all part types in category C are averaged to get mIoU for that shape. To calculate mIoU for the category, we take average of mIoUs for all shapes in that category.

In this section, we compare our segmentation version PCAPN (a modified version of Table 1, Segmentation Network) with two traditional methods [34], [35] that both take advantage of point-wise geometry features and correspondences between shapes, as well as our own 3D CNN baseline.

B. NETWORK MODEL ANALYSIS

In this section, two sets of experiments were carried out to demonstrate the advantages of our model. The first set of experiments is to compare the accuracy of some of the mainstream network models retained in the training model. In the second experiment, the training time of the PointNet++ network model and the PCAPN network model were compared while the sampling algorithm was unchanged. In all the experiments, we selected the internationally-recognized ModelNet40 datasets as our main training and testing dataset for this experiment.

On one hand, the first set of experiments. Table 2 lists a number of popular point cloud classification algorithms (e.g. PointNet, MVCNN [38], PointNet++, SO-Net [8], PAT [39], etc.). From the table, we can see that our PCAPN algorithm has certain advantages in accuracy, which is 2.9% higher than PointNet, the pioneering work of point cloud classification and 1.1% than higher the optimized version of PointNet++. Although the improvement is not that significant points, it is important to know that the more accurate the model is, the more difficult it can be improved. Even a few percent improvement is a qualitative leap in performance. Many of the later algorithm improvements remain at a few percentage points. So this also has a good meaning for our PCAPN algorithm. From the point of view of calculation time, we only compare PointNet++ with our PCAPN algorithm here. As shown in Table 3, our PCAPN algorithm is superior to PointNet++ algorithm in training time. The main reason is that our network structure and our sampling algorithm sort and classify all points from the very beginning. It is important to do this only once, and the complexity of the FPS algorithm we have analyzed above is $o(N^2)$, which is squared with the number of points, greatly increasing the computational complexity. Moreover, our network structure can reuse the ordered point set, which also reduces the computational time to some extent and avoids multiple ordering of the point set. Another interesting thing is that our network had an accuracy of very close to 90% at the beginning of the training. This might be due to the special relationship between our sampling algorithm and the feature of the object represented by the 3D point cloud.

TABLE 1. Segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points. We compare with two traditional methods [36] and [37] and a 3D fully convolutional network baseline proposed by us. Our PCAPN method achieved the state-of-the-art in mIoU.

Method	mean	aero	bag	cap	car	chair	phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
#shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
Wu	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
Yi	81.4	**81.8**	**78.4**	**77.7**	**75.7**	87.6	61.9	**92.0**	85.4	**82..5**	**95.7**	**70.6**	91.9	**85.9**	**53.1**	69.8	75.3
3DCNN	79.4	75.1	72.8	73.3	70.0	87.3	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Ours	**82.7**	79.5	69.5	**78.0**	70.5	**88.8**	**73.6**	90.0	**85.9**	81.8	94.8	59.5	**93.5**	78.4	51.0	**73.0**	**81.1**

Besides, for the proposed two sampling algorithms (APS and MPS), it is not difficult to see from the table that the APS method has an advantage over MPS in terms of accuracy. The reason may be that the APS method extracts points from each level. This maximizes the coverage of points in all areas, extracts more features, and has slightly better accuracy. While the MPS method extracts the most important points in the point set, such as inflection points, it is likely to ignore some other secondarily important points, which is a big loss for us to extract features. In other words, although the APS method may miss a small number of important points, it can extract the features of some minor points, which ensures the comprehensiveness of our feature extraction. While the MPS method extracts more features of important points than the APS method, it misses some feature information of secondarily important points. Moreover, in many cases the MPS may be saturated with the feature information of important points. This may also be the reason why APS has an advantage over MPS in accuracy.

On the other hand, the other set of experiment, which replaces the APS and the MPS algorithms in our network model with the FPS, and compares them with our PCAPN algorithm in terms of accuracy and computational time. Table 2 and Table 3 show when the best models appear and their accuracy. It is not difficult to see from the table that using the FPS method is better than using MPS method and PointNet++, but less accurate than the APS algorithm. This also demonstrates that our network model has certain advantages. From the point of view of computational time, the APS sampling method and MPS method can produce the optimal model earlier than the FPS method, which further confirms that our APS and MPS methods have some advantages in computational time.

TABLE 2. Current main algorithms for three-dimensional point clouds and their accuracy(left column as method, right column as accuracy, counted by percentage).

Method	Acc%
PointNet(Vanilla)	87.2
Pointnet	89.2
MVCNN	90.1
PointNet++	90.7
SO-Net	90.9
RGCNN	90.5
PAT	91.7
SpecGCN	91.5
PCAPN (MPS)	**90.7**
PCAPN (APS)	**91.8**

TABLE 3. Column 2 compares the time required for the PointNet++ and Point PCA methods to achieve 90% accuracy. The third column compares how long it takes these three algorithms to produce the best model. The fourth column compares the accuracy of the best models. (All the time here is a rough number, not a specific time. There may be varying degrees of training between different machines, time units are hours).

Method	Time with 90% Acc(h)	Time with Best Acc(h)	Acc%
PointNet++	14	18	90.7
PCAPN(MPS)	**12**	**13**	**90.7**
PCAPN (APS)	**12**	**13**	**91.8**

C. ANALYSIS OF SAMPLING METHODS

To further verify the feasibility of our proposed APS and MPS sampling methods, we did a comparative experiment with PointNet++ network model and FPS using the control variable method. In this experiment, we selected internationally-recognized ModelNet40 (40 categories of CAD models, 9843 shapes for training, and 2468 shapes for testing) as our main training and testing dataset for this experiment. The experiment, consists of three steps. Firstly, ModelNet40 dataset was trained using the PointNet++ network model (the FPS method used in PointNet++), the training result with accuracy reaching 90% and the final accuracy was record. Secondly, while keeping the other conditions unchanged, all the places of PointNet++ that use the FPS algorithm were replaced with APS and MPS sampling methods, and the training results with 90% accuracy and final accuracy were recorded. Thirdly, the feasibility of our sampling method was proved by comparing the advantages and disadvantages of APS, MPS and FPS for these three sets of data analysis.

As shown in Fig. 4, it is clear that when the training accuracy is more than 90%, APS reduces training time a lot compared to the FPS in the same configuration on the computer. One interesting thing to discover is that only two batches have been trained with our APS. This also reflects from the side that our APS is less computationally complex

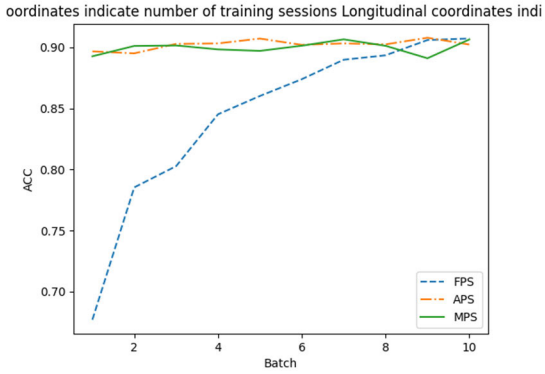


FIGURE 4. Accuracy of three sampling algorithms training in PointNet++ model.

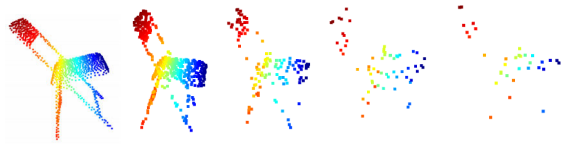


FIGURE 5. APS (number of sample points from left to right is 1024, 512, 128, 64, and 32, respectively).

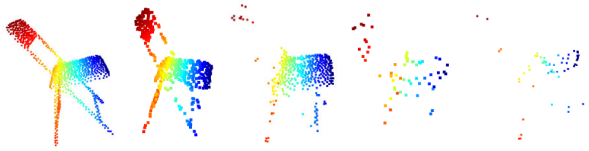


FIGURE 6. MPS (number of sample points from left to right is 1024, 512, 128, 64, and 32, respectively).

FIGURE 7. Farthest point sampling (number of points from left to right is 1024, 512, 128, 64, and 32, respectively).

FIGURE 8. Random sampling (number of points from left to right is 1024, 512, 128, 64, and 32, respectively).

than the FPS. Under the same conditions, APS takes precedence over the FPS for more than 90% accuracy, and the final training results are close to 90.7%, which is very close to the training results of PointNet++. To sum up, our APS is less computationally expensive than the FPS and has little difference in coverage.

Through this comparative experiment, it can be found that our APS and MPS algorithms are feasible, and also have some advantages in time, with 90% accuracy over the FPS. Furthermore, our sampling algorithm reached 90% accuracy quickly over a period of training and tended to be stable, with little variation around the 90% accuracy. It is plausible that to some extent, the points we have extracted are likely to be representative; to a large extent, this portion of the sample points can be used to approximate our original input sample points.

D. VISUALIZING

In this section, we focus on our sampling methods, as well as farthest point and random sampling. We visualized FPS, RS, APS, and MPS.

Here, we selected 1024 points, 512 points, 128 points, 64 points and 32 points from RS, FPS, APS and MPS respectively. As shown in Fig. 5, 6, 7, 8, when we selected 1024 points and 512 points, the results of RS, FPS, APS, and MPS methods still indicate that this is a chair. However, when we continued to reduce the number of sampling points, such as 128 points, 64 points or even fewer, random sampling hardly recognized what the point cloud representd. To say 32 points without exaggeration, you only know that it is a point. On the contrary, the most FPS, APS and MPS can still recognize this chair at 128 points and 64 points. When the number of points was reduced to 32 points, only the outline of the chair.

Here is the possible reason. As we all know, random sampling is to take some units from the whole unit as samples according to the principle of randomness. Here we randomly select a part of points **{xi, xi+1,…, xi+m}** from the input point **{x1, x2,…, xn} (where xi, xi+m <{x1, x2,…, xn})**, which shows that the selection of sampling points is random, and when we select more sampling points, we will be very close to the input point. However, as the number of sampling points decreases and the sampling points are random, they will gradually deviate from the input points or even be completely unrecognized. This confirms the phenomenon in our previous experiment that, why random sampling identifies a chair when 1024 points, 512 points and 128 points are selected. However, when 64 points and 32 points are selected, it is not even recognizable at all, knowing that this is a bunch of points; FPS calculates the Euclidean distance between points and calculates the farthest point from the previous point. Thus, FPS also has a high coverage, and the points sampled can be outlined. Our sampling method is to calculate the contribution ratio of each point, sort the original set of input points by the contribution ratio, select the medium difference or select the main points; Therefore, the point coverage is not inferior to the FPS method. The most important thing is that this method can control the points sampled without repeating the selection, which is more suitable for our network model.

Conclusion: in this work, we propose multi-scale point, a convolutional neural network algorithm for processing point cloud in metric space. In order to reduce computational

redundancy and reduce the probability of sampling to duplicate points, we propose two new sampling methods, namely APS and MPS. In order to extract more feature information of the input point cloud, we use a multi-scale hierarchical convolution neural network to extract the feature information of the point cloud, and splice and integrate the feature information extracted at each scale. It is not difficult to see from our experiments that our network has obvious advantages in accuracy, mIoU and efficiency.

V. SUMMARY

In this algorithm, we propose a multi-scale point network model and two sampling algorithms, namely APS algorithm and MPS algorithm. They have certain advantages in accuracy and reducing computational redundancy. Firstly, we analyze the advantages of PointNet and PointNet++ and the problems of these two network models. Then we propose our solution. Finally, we evaluate the advantages of our network model by comparing mIoU, accuracy and network model performance. Our network model improves the accuracy by 1.1% to 91.8% based on the 90.7% accuracy of PointNet++, which exceeds state-of-the-art many network models or algorithms about 3D point cloud. In order to further verify the superiority of the proposed algorithm, experiments were carried out. The results were evaluated from the aspects of efficiency, sampling method and training accuracy to prove the feasibility and superiority of the proposed algorithm.

ACKNOWLEDGMENT

The authors would like to thank the school of communication engineering of Chengdu University of information engineering for its strong support They would also like to thank the tutor and team for their hard work.

COMPLIANCE WITH ETHICAL STANDARDS

Conflict of interest. The authors declares that they have no conflict of interest.

Declaration of competing interest The authors have no competing interests to declare that are relevant to the content of this paper.

REFERENCES

[1] Y. Zhuang, Y. Liu, G. He, and W. Wang, "Contextual classification of 3D laser points with conditional random fields in urban environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Hamburg, Germany, Oct. 2015, pp. 3908–3913.

[2] Y. Lu and C. Rasmussen, "Simplified Markov random fields for efficient semantic labeling of 3D point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 2690–2697.

[3] G. M. Weiss, K. Yoneda, and T. Hayajneh, "Smartphone and smartwatch-based biometrics using activities of daily living," *IEEE Access*, vol. 7, pp. 133190–133202, 2019.

[4] G. David and B. Jan, "A review on deep learning techniques for 3D sensed data classification," *Remote Sens.*, vol. 11, no. 12, p. 1499, 2019.

[5] Y. Xie, J. Tian, and X. Xiang Zhu, "Linking points with labels in 3D: A review of point cloud semantic segmentation," 2019, *arXiv:1908.08854*.

[6] J. Zhang, X. Zhao, and Z. Chen, "Review of semantic segmentation of point cloud based on deep learning," *Laser Optoelectron. Prog.*, vol. 57, no. 4, pp. 20–38, 2020.

[7] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," 2019, *arXiv:1912.12033*.

[8] J. Li, B. Chen, and G. Lee, "So-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 9397–9406.

[9] C. R. Qi, H. Su, and K. Mo, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017, pp. 652–660.

[10] C. R. Qi, L. Yi, and H. Su, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 5099–5108.

[11] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *Proc. Symp. Geometry Process.*, vol. 6, 2003, pp. 156–164.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] Q. Lu, C. Chen, W. Xie, and Y. Luo, "PointNGCNN: Deep convolutional networks on 3D point clouds with neighborhood graph filters," *Comput. Graph.*, vol. 86, pp. 42–51, Feb. 2020.

[14] Y. Li, R. Bu, M. Sun, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Montreal, BC, Canada, 2018, pp. 828–838.

[15] W. Wu, Z. Qi, and F. Li, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Seoul, South Korea, Jun. 2019, pp. 9621–9630.

[16] H. Zhao, L. Jiang, C. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, Jun. 2019, pp. 5565–5573.

[17] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 11105–11114.

[18] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[19] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2015, pp. 922–928, doi: 10.1109/IROS.2015.7353481.

[20] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2016, pp. 5648–5656, doi: 10.1109/CVPR.2016.609.

[21] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: Field probing neural networks for 3D data," 2016, *arXiv:1605.06240*.

[22] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Robotics: Science and Systems*, Rome, Italy, 2015, p. 1317.

[23] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3D deep shape descriptor," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2319–2328, doi: 10.1109/CVPR.2015.7298845.

[24] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "BlenSor: Blender sensor simulation toolbox," in *Proc. Int. Symp. Vis. Comput.* Berlin, Germany: Springer., 2011, pp. 199–208.

[25] X. Xu, C. Liu, and Y. Zheng, "3D tooth segmentation and labeling using deep convolutional neural networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 7, pp. 2336–2348, Jul. 2019, doi: 10.1109/TVCG.2018.2839685.

[26] D. Ekiz, Y. Said Can, Y. Ceren Dardagan, and C. Ersoy, "Can a smartband be used for continuous implicit authentication in real life," *IEEE Access*, vol. 8, pp. 59402–59411, 2020.

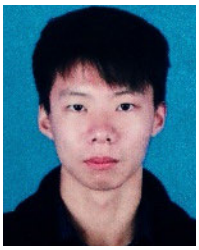
[27] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," 2015, *arXiv:1511.06391*.

[28] K. Sun, L. Wang, B. Xu, W. Zhao, S. Wei Teng, and F. Xia, "Network representation learning: From traditional feature learning to deep learning," *IEEE Access*, vol. 8, pp. 205600–205617, 2020, doi: 10.1109/ACCESS.2020.3037118.

[29] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," 2016, *arXiv:1611.05009*.

[30] A. A. M. Muzahid, W. Wan, F. Sohel, N. U. Khan, O. D. Cervantes Villagómez, and H. Ullah, "3D object classification using a volumetric deep neural network: An efficient octree guided auxiliary learning approach," *IEEE Access*, vol. 8, pp. 23802–23816, 2020, doi: 10.1109/ACCESS.2020.2968506.

[31] Z. Wu, R. Shou, Y. Wang, and X. Liu, "Interactive shape co-segmentation via label propagation," in *Comput. Graph.*, vol. 38, pp. 248–254, Feb. 2014.
[32] A. Qi, J. Wei, and B. Bai, "Research on deep learning expression recognition algorithm based on multi-model fusion," in *Proc. Int. Conf. Mach. Learn., Big Data Bus. Intell. (MLBDBI)*, Nov. 2019, pp. 288–291, doi: 10.1109/MLBDBI48998.2019.00064.
[33] Y. Seo and K.-S. Shin, "Image classification of fine-grained fashion image based on style using pre-trained convolutional neural network," in *Proc. IEEE 3rd Int. Conf. Big Data Anal. (ICBDA)*, Mar. 2018, pp. 387–390, doi: 10.1109/ICBDA.2018.8367713.
[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[35] L. Yi, V. G. Kim, D. Ceylan, I. C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, p. 210, 2016.
[36] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.
[37] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," 2016, *arXiv:1612.00606*.
[38] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953, doi: 10.1109/ICCV.2015.114.
[39] G. Te, W. Hu, A. Zheng, and Z. Guo, "RGCNN: Regularized graph CNN for point cloud segmentation," in *Proc. 26th ACM Int. Conf. Multimedia*, Seoul, South Korea, 2018, pp. 746–754.



YULIN JI received the bachelor's degree from Qufu Normal University, in 2018. He is currently pursuing the master's degree with the Chengdu University of Information Engineering, China. He has worked at Inspur, in 2018. Since 2020, he has mainly studied artificial intelligence and computer vision, as well as 3D point cloud and deep learning algorithm at the Computer Communication Research Laboratory, Chengdu University of Information Engineering. He has published two journal articles.

JIANDAN ZHONG received the Doctorate degree in signal and information processing from the University of Electronic Science and Technology, in 2018. He has successively worked at Tieto, MediaTek, Institute of Optoelectronic Technology, Chinese Academy of Sciences, Chengdu University of Information Engineering, and other units. He has rich experience in scientific research and engineering development. He is a Doctor. He is currently a Lecturer at the School of Communication Engineering, Chengdu University of Information Engineering; and a part-time Postdoctoral Researcher at the National Key Laboratory of Communication Anti-Interference Technology, University of Electronic Science and Technology. He has published 13 scientific research papers in important academic journals and conferences at home and abroad, including seven SCI searches and five EI searches. He is mainly engaged in a number of national, provincial, and horizontal cooperation projects. He is a member of the China Communication Society and the China Artificial Intelligence Youth Trade Union. He has served as a Reviewer for international SCI journals, such as *Remote Sensing* and *ISPRS International Journal of Geo Information and Dragons*.

YINGXIANG LI was born in November 1972. He received the Ph.D. degree in communications and information systems from the University of Electronic Science and Technology, in March 2003. Later, he worked at TCL Mobile Communications Company Ltd., Sichuan Communications Research Planning and Design Company Ltd., Chengdu Information Engineering University, and other units. He served as the Deputy Director General for the Dual Stream Investment Promotion Bureau, in 2012 and Deyang Investment Promotion Bureau, in 2014. In December 2012, the University of Electronic Science and Technology and Chengdu High-Tech Zone jointly trained postdoctoral students. Since December 2012, he has served as the Vice President, the Vice President (presiding over work), and the President for the Institute of Communications Engineering. He has mainly engaged in edge computing, artificial intelligence, SOC and on-chip systems, communication terminals, Internet of Things technology, weak signal processing, and other fields.

He is a Professor, a member of 93 Society, a Senior Member of the China Communications Association, a member of the Expert Library of Sichuan Communications Association, the Director of the Chengdu Integrated Circuit Industry Association, the Director of Sichuan Intelligent Communications Terminal Industry Alliance, and the Director of Beidou Industry Alliance of Sichuan. He has published more than 50 scientific papers, of which eight were retrieved by SCI and 20 by EI. He has been granted 36 patent rights for utility models, 12 software copyrights, more than ten patent applications for inventions and eight authorizations. In recent years, he has hosted or participated in about 20 national, provincial, and municipal vertical and enterprise horizontal projects. He is the Editor-in-Charge of *Innovation Practice Textbook*.

JUNJIE FU graduated from the Chengdu College of University of Electronic Science and Technology, in 2019. He is currently pursuing the Postgraduate degree in information engineering with the Chengdu University of Information Engineering. He mainly studies artificial intelligence and computer vision, as well as 3D point cloud and depth learning algorithm.

JIAWEI LIU graduated from Chengdu University, in 2020. He is currently pursuing the Postgraduate degree in information engineering with the Chengdu University of Information Engineering. He mainly studies artificial intelligence and computer vision, as well as 3D point cloud and depth learning algorithm.