IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 26, 2019, accepted March 30, 2019, date of publication April 9, 2019, date of current version May 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909742

3D Point Cloud Analysis and Classification in Large-Scale Scene Based on Deep Learning

LEI WANG[1,2,3], WEILIANG MENG[4,5], RUNPING XI[1,2], YANNING ZHANG[1,2], CHENGCHENG MA[4,5], LING LU[3], AND XIAOPENG ZHANG[4,5]

[1]School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China
[2]National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, Xi'an 710072, China
[3]Jiangxi Engineering Laboratory on Radioactive Geoscience and Big Data Technology, East China University of Technology, Nanchang 330013, China
[4]LIAMA–NLPR, CAS Institute of Automation, Beijing 100190, China
[5]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding authors: Lei Wang (wlei598@163.com), Runping Xi (xrp@163.com), and Yanning Zhang (ynzhangnpu@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61561003, Grant 61571439, Grant 61572405, Grant 61761003, and Grant 61571046, and in part by the Beijing Natural Science Foundation under Grant 4184102 and Grant L182059.

ABSTRACT We present a deep learning framework for efficient large-scale 3D point cloud analysis and classification using the designed feature description matrix (FDM). As the 3D points are unordered in the large-scale scene, and no topology structure can be employed directly for classification and recognition, it is difficult to apply deep neural network directly on 3D point clouds as points cannot be arranged in a fixed order as 2D image pixels. We design a new pipeline for 3D data processing by combining the traditional features extraction method and deep learning method. Our FDM encapsulates the 3D features of the point and can be used as the input of the deep neural network for training and testing. The experiments demonstrate that our method can acquire higher classification accuracy compared with our previous work and other state-of-art works.

INDEX TERMS CNN, feature description matrix, geometric features, point cloud.

I. INTRODUCTION

3D Point cloud classification plays an important role in lots of applications such as remote sensing and scene reconstruction. As obtaining 3D point clouds from the real scene becomes cheap, convenient and fast, point clouds are ubiquitous and the automatic classification of them will save a lot of time and cost. In the point cloud, points are independent with each other and no connections information can be employed, making the semantic classification information difficult to be inferred directly. A lot of traditional point cloud classification methods have been proposed to extract the geometric features of each point based on their local neighborhood. The geometric properties of natural surfaces may span over a wide range of scales (from *cm* to *km*), and lots of works have been done on the natural scenes understanding such as dune fields [1] and on the urban scenes [2]. These methods achieve good classification results, but the precisions still need further improvement.

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

In recent years, deep learning gains a lot of attentions because of its excellent performance, especially on image recognition and understanding. It can construct complex connections between input images and corresponding labels by neural networks in turn to recognize testing images superior to humans. Deep learning cannot be directly applied to 3D information classification, because the order of point in the point cloud is unrelated to the structure of the scene, making most neural networks failed on feature extraction and point classification. If some features that unrelated to the point orders and point translations can be extracted explicitly for neural networks, a better classification result may be obtained. Based on this idea, we propose a new point cloud classification framework which combines traditional feature-based methods with deep learning methods. We first extracted a series of features for each point based on their optimal neighborhood, and construct the Feature Description Matrix (FDM). FDMs are further input to convolution neural network to obtain invincible classification results. The key point is that geometric features can be extracted in advance which convolution neural network cannot learn directly, and

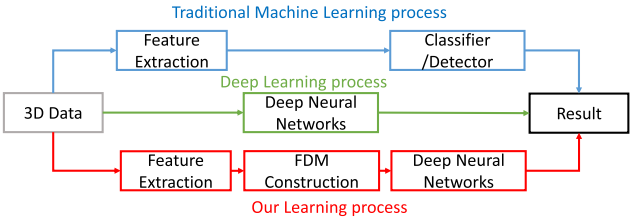

FIGURE 1. Processing comparison.

then deep learning network can be trained better based on these features than just from the original information. This work is extended from our work [3], and we give more deep learning methods on our framework to validate our method. Two more features including the local point density and the nearest neighbor quadrihedron volume are encapsulated in the FDM. Guidelines for the training are also be proposed to improve the final classification accuracy. Figure 1 shows our learning process compared with previous methods, and Figure 2 shows the architecture of our algorithm, in which CNN are used as the representative deep learning method.

The contributions are listed as follows:

- A new pipeline for 3D data processing by combining the traditional feature extraction methods and deep learning methods. We first extract the features from the 3D points explicitly which are usually cannot be 'seen' directly by neural networks, and classify the points based on these features based on deep learning methods.
- A series of guidelines on different parameters on deep learning networks for FDMs training and test. We use different settings to train and test FDMs including the sample enhancement, the loss function, the optimizer, the feature arrangement, and the learning rate.

Our algorithm can effectively classify the point cloud into different categories with higher accuracies after the training process(see Table 5). Based on our work, more deep learning networks can be applied based on our guidelines.

II. RELATED WORKS

In order to classify a point cloud, traditional classification methods use hand-crafted features [4], [5], contextual features [6], and specific color, shape or geometry features [7] as the basis. Chehata *et al.* [8] classified point clouds by using random forests with 21 features. Guo *et al.* [9] utilized JointBoost with 26 features to classify point clouds into five classes. Kragh *et al.* [10] used the SVM classifier with 13 features to classify point clouds. Brodu and Lague [11] extracted multiscale features from different neighborhoods for classification. Weinmann *et al.* [12] divided machine learning methods of point cloud classification into 6 categories including instance-based learning [13], rule-based learning [14], probabilistic learning [15], Max-margin learning [16], ensemble learning [17] and deep learning [18], while they presented a new, fully automated and versatile framework composed of four components, and demonstrated that the selection of optimal neighborhood for the 3D point can significantly improve the results of 3D scene analysis.

Recently, the deep-learning technique can automatically and jointly learn the features and classifiers from the multiple types of data [19]–[26] including images, videos, speeches, and audios. It has a wide range of applications, such as hyperspectral image classification, handwritten digit recognition, underwater images enhance and face detection etc. Deep learning as one of artificial intelligence technique [27] has a great success in image classification and recognition, some people try to apply it to 3D data. Many research works in remote-sensing scene understanding has focused on learning feature representations using deep learning frame network. Chen *et al.* [28] proposed a 3DCNN based feature extraction model with combined regularization to extract effective spectral-spatial features of hyperspectral imagery. Zhang *et al.* [29] provided a general framework of Deep Learning for RS data combined with various deep networks and tuning tricks.

Meanwhile, a series of neural network methods for 3D shapes are proposed. Koppula *et al.* [30] used depth images with different perspectives of 3D objects as the input, and utilized auto-encoder with pre-training DBN to extract features. Wu *et al.* [31] designed a volumetric CNN architecture on 3D voxel grids to represent a geometric 3D shape for object classification and retrieval. For point clouds, Huang and You [32] turned the point cloud into 3D voxels and use occupied voxels to feed the 3D convolutional layer for training. Engelcke *et al.* [33] discretized the point cloud into a sparse 3D grid, extracted the statistics number of the cells, and performed a sparse convolution by voting without considering the more low-level input representations. Liu *et al.* [34] provided a 3DCNN-DQN-RNN method which combines the 3D convolutional neural network (CNN), Deep Q-Network(DQN) and Residual recurrent neural network (RNN) together for an efficient semantic parsing of large-scale 3D point clouds. Alexandre *et al.* [35] transformed the point cloud into RGB and depth image views for CNN training. Qi *et al.* [36] gave the 'PointNet' for object classification and scene semantic parsing, in which point coordinates are input to the neural network directly. However, as the local structures of the metric space cannot be captured, it is difficult to recognize fine-grained patterns on complex scenes. The PointNet++ [37] used hierarchical (or multiscale) neural network to solve this problem, but applying it on the large-scale 3D point clouds is unfeasible as the scene usually includes the solid points as well as the surface points. Other deep learning methods are developed for shape segmentation. Guo *et al.* [38] proposed a novel approach for labeling on 3D meshes by using Deep Convolutional Neural Networks, while Luciano and Hamza [39] gave an integrated framework for 3D shape classification using deep learning with geodesic moments.

These methods mainly for shape classification and segmentation but not for the large scene point cloud

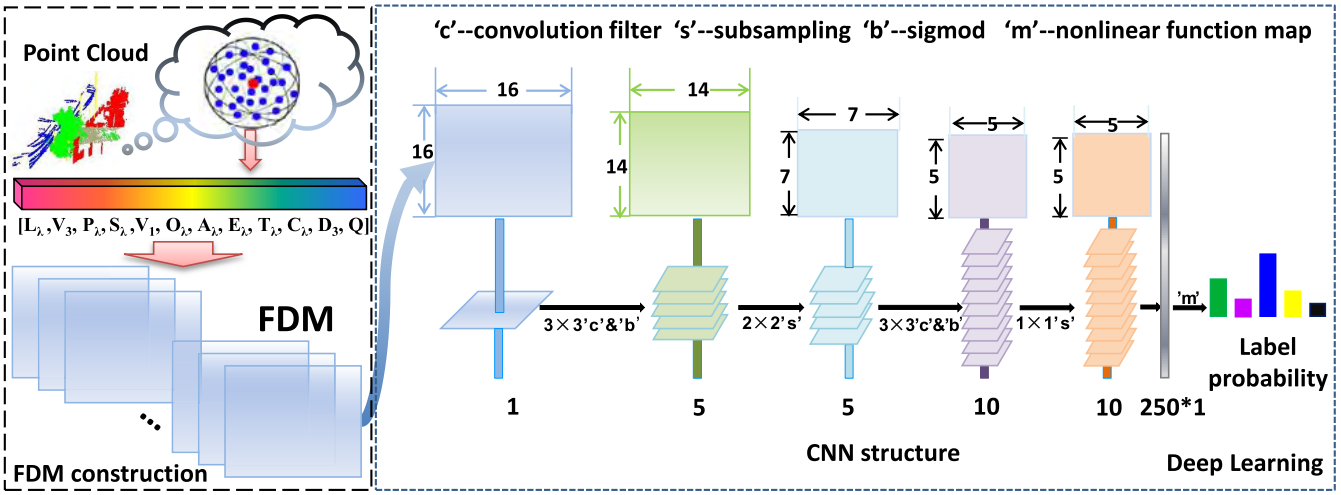

FIGURE 2. The architecture of our algorithm. After constructing the *FDM*, here we use a '5c-2s-10c-1s' CNN for training and testing. Note that we use sigmoid function but not the tanh function as it works better in our experiments, although tanh function is better than sigmoid usually in image classification field.

classification, as most methods need the topology information for the calculation. However, the potential application of point-cloud classification is still relatively unexplored based on deep learning. The main difficulty in classifying points by deep learning stems from the unorganized point clouds. There is an obvious gap between deep learning and large-scale 3D point clouds.

III. ALGORITHM

The architecture of our algorithm is given in Figure 2. Note that coordinates x, y, z are not the decisive factors for the classification. Specifically, if we move or rotate an object to a new position, the coordinates are changed but it still belongs to the same class. The other difficulty for deep learning on 3D points is the irregular arrangement in the point cloud, meaning that different order of the points doesn't change the point cloud while this may disturb the deep learning process. Besides, the classification of each point in the point cloud is mainly determined by its local neighborhood features, while these features cannot be learned directly by the deep learning methods. Here we provide an available algorithm for classification of point cloud based on 3D features and deep learning network. In order to identify the class of each point in the large-scale scene point cloud, we first extract the features from the point cloud explicitly, then we train a neural network for classification. Our work is different from the work [38], as the method of [38] can only deal with meshes but not the point cloud.

The features of each point in the point cloud are extracted from its optimal neighborhood, then Feature Description Matrix (FDM) for the point can be constructed to input the neural network. The following denotations are used for clear description. Given a point cloud with N points, the i-th point P_i has the coordinates (x_i, y_i, z_i). The features of P_i are extracted from its optimal neighborhood containing P_i and its k nearest neighbors, where $k \in R$ is an integer that can minimize the eigenentropy of P_i's neighborhood. The coordinates of all points in the optimal neighborhood are denoted as a $(k+1) * 3$ matrix M, and the corresponding covariance matrix is denoted as C with size $3 * 3$. C is a symmetric positive-definite matrix, and its three eigenvalues are larger than 0, denoted as λ_1, λ_2, and $\lambda_3 \in R$ respectively where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. The eigenvectors of C are V_1, V_2, and V_3, corresponding to λ_1, λ_2, and λ_3 respectively. The normalized eigenvalues are $e_i = \lambda_i / \Sigma_{\lambda_i}$ for $i \in \{1, 2, 3\}$.

A. OPTIMAL NEIGHBORHOOD SELECTION

The optimal neighborhood should be the 'best' neighborhood of the point which can reflect the local geometry features centered on this point. In order to search the most representative neighborhood, we employ the method [40] to minimize the eigenentropy of Eq 1 for the optimal neighborhood selection, as the eigenentropy provides a measure of the order of 3D points within the covariance ellipsoid. We choose all the integer values in $[k_{min}, k_{max}]$ with $k_{min} = 10$ and $k_{max} = 100$, and get the optimal neighborhood with minimum E_λ. Note that each point may have different point number in its optimal neighborhood.

$$E_\lambda = -e_1 ln(e_1) - e_2 ln(e_2) - e_3 ln(e_3) \tag{1}$$

B. 3D FEATURES EXTRACTION

After obtaining optimal neighborhood for each point, we extract 3D features which should be invariant from translation. We divided these features into eigenvalue-based features and geometric features. Here we give more details on these features.

1) EIGENVALUE-BASED FEATURES

The eigenvalue-based 3D features are calculated from the corresponding eigenvalues of M as follows:

- Linearity: $L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} = \frac{e_1 - e_2}{e_1}$
- Planarity: $P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} = \frac{e_2 - e_3}{e_1}$
- Scattering: $S_\lambda = \frac{\lambda_3}{\lambda_1} = \frac{e_3}{e_1}$

- Omnivariance: $O_\lambda = 3\sqrt[3]{e_1 e_2 e_3}$
- Anisotropy: $A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1} = \frac{e_1 - e_3}{e_1}$
- Eigenentropy: $E_\lambda = -\sum_{i=1}^{3} e_i ln(e_i)$
- Trace: $T_\lambda = \frac{2}{\pi} \arctan(\lambda_1 + \lambda_2 + \lambda_3)$
- Change of curvature: $C_\lambda = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} = 3e_3$

Almost all these features are in [0, 1] except E_λ, and most of the features can be found in [40]. As E_λ is used for the optimal neighborhood selection, it is important and use the original value will be superior to its normalized value for the classification, which is validated by our experiments. If we increase the weight of E_λ properly, the classification accuracy is also be improved, but if the weight is too large, the classification accuracy will degrade.

Compared to [40], we also add the coefficient 3.0 on O_λ in the Omnivariance and Change of Curvature features for normalization, which can improve the classification accuracy. Besides, [40] use the sum of eigenvalues, while in our case we use Trace. As $e_1 + e_2 + e_3 = 1.0$ after the normalization which is not significant for the classification, and $\lambda_1 + \lambda_2 + \lambda_3 \in (0, \infty)$ in theoretical, we design the Trace feature by normalizing $\lambda_1 + \lambda_2 + \lambda_3$ into (0, 1) in turn to promote the classification accuracy.

These features have obvious geometry meanings, and here we give some plain explanations. Linearity L_λ reflects the linearity of the point, and if the neighborhood points are in a line, L_λ gets close to 1.0, and Anisotropy A_λ is 1 as e_1 is 1 and e_3 is 0. For the plane case, e_1 and e_2 are nearly the same with value 0.5 and e_3 is nearly zero, which makes Planarity P_λ close to 1.0. For the sphere case, e_1, e_2, and e_3 are equal, meaning Scattering S_λ of the points are uniform, i.e, $S_\lambda = 1.0$, Omnivariance O_λ reaches the biggest value 1.0, and Anisotropy A_λ is 0 as $e_1 = e_2 = e_3$. If the difference of e_1 and e_3 is large, Anisotropy A_λ will be large. The sum of λ_1, λ_2, and λ_3 is the Trace of C which reflects the invariant point number based on the eigenvectors of C, and we normalize it into [0,1] based on arctan function for better classification validated by our test. The Change of Curvature C_λ introduced by Pauly *et al.* [41] describes the variation along the surface normal of P_i, i.e., it estimates how much the points deviate from the tangent plane. We add the coefficients 3.0 compared to [41]. For completely isotropically distributed points such as planes, $C_\lambda = 0$.

Although these features are categorized as eigenvalue-based features but not geometric features, they have great relations with the geometric property of the point essentially. Now we can obtain 8 eigenvalue-based features for each point, but just using this features for classification cannot achieve a robust result. We also need to employ other 3D features.

2) GEOMETRIC FEATURES

Three eigenvectors of C are geometric 3D features, where V_1 represents the maximum distribution direction of the points, V_3 represents the minimum distribution direction of the points and V_2 represents the distribution direction that is perpendicular to V_1 and V_3. If the point P_i's optimal neighborhood is sampled from a surface, then V_1 and V_2 are the two principle directions on P_i for the surface respectively, while V_3 is the normal direction of the surface on P_i; if P_i's optimal neighborhood is sampled from a line, V_1 is the direction of the line; if P_i's optimal neighborhood is sampled from a plane, V_3 is the normal direction of the plane.

Compared to our previous work [3], we also use two new features as the geometric features: local point density and nearest neighbor quadrihedron volume. The local point density D_3 of P_i in 3D given by Weinmann *et al.* [42]:

$$D_3 = \frac{k+1}{\frac{4}{3}\pi r_{kNN}^3} \tag{2}$$

Here r_{kNN} is the maximum distance from the point to its k-Nearest Neighborhood (kNN) points in the optimal neighborhood.

The nearest neighbor quadrihedron volume Q of vertex P_i can be calculated as the mixed product of the three vectors constructed by three nearest neighbor P_{1_i}, P_{2_i}, P_{3_i} of the vertex P_i as in [43]:

$$Q = \frac{1}{6} \| (\overrightarrow{P_i P_{1_i}} \times \overrightarrow{P_i P_{2_i}}) \cdot \overrightarrow{P_i P_{3_i}} \| \tag{3}$$

C. FEATURE DESCRIPTION MATRIX CONSTRUCTION

Usually, the class label of the point usually has some connections with its features. Not all of the features are necessary for the classification. The label of the point is usually based on the geometry related properties, i.e, the points in the line and the points on the plane should have different class labels. This is natural, as the 3D objects with the same label usually have the same shape. However, if we just use the eigenvalues or eigenvectors for the training, the classification result is bad. In order to classify the points effectively, we should organize the extracted features efficaciously to feed the neural network.

1) FEATURE DESCRIPTION VECTOR

For each point P_i, we construct a Feature Description Vector (FDV), in which the element should be invariant to translations, as the same as the label of the point. Many features can be selected as the element of the FDV, but not all the options are effective. We have tested many cases, and give a stable option as $fv_i = [L_\lambda, V_3, P_\lambda, S_\lambda, V_1, O_\lambda, A_\lambda, E_\lambda, T_\lambda, C_\lambda, D_3, Q]$. Note that V_1 and V_3 are vectors with 3 elements, and the length of fv_i is 16. The arrangement order for each element in fv_i can be varied and will have some impacts on the final classification result, but in general, the overall accuracies are close for different arrangements. We have tested different orders and current arrangement can bring up a higher classification accuracy.

2) FEATURE DESCRIPTION MATRIX

The Feature Description Matrix is constructed based on the above FDV for each point. We choose the 15-nearest neighbors of P_i and itself use the corresponding FDVs with P_i FDV

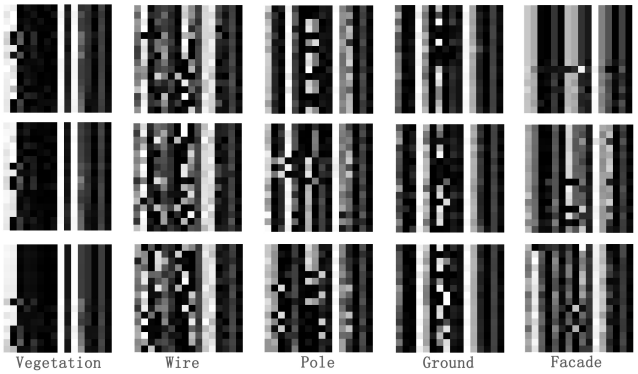

FIGURE 3. FDMs of different points in different categories. We randomly choose 3 points from each category in the training data (Figure 6(a)) and transform their FDMs to images for better observation. We can see that the same category points have similar FDMs visually.

to construct the Feature Description Matrix(or FDM) with size of $16 * 16$. We make FDM be a square matrix as this is more stable. Theoretically, the number of nearest neighbors k (here is 15) should not be too less or too more, as the values in each FDV are based on the optimal neighborhood containing k_o points. The ideal state is $k = k_o$. In practice, our option works well in tests, and Figure 3 shows the FDMs of different points from different categories, and we can see that the same category points have the similar FDMs. These similarities can be captured by CNN for classification.

D. TRAINING AND TESTING

As the FDM is very similar to an image, it can be used to input a neural network in order to construct the mapping between the FDM and the corresponding label of a point in the point cloud. For the training data, we use their FDMs and labels to train the CNN to get the parameters, then apply these parameters to classify the testing data based on their extracted FDMs. The pseudocode of our algorithm is given in Algorithm 1. The classification goes through two feature extraction pass essentially. The first pass is to extract FDM explicitly for every point, and the second pass is done by neural network implicitly. Our method integrates the previous classification methods and can achieve a better classification result.

Our CNN for the point cloud classification is inspired by LeNet-5 and is shown in Figure 2. We use 5 convolution kernels and 2 times subsample, then 10 convolution kernels and no downsampling for the output. The kernels size is 3*3. We use 50 FDMs to construct one batch, and after 30 iterations (pointed out by [3]), the batch errors tend to be stable, and we set 30 as the default training times.

Our experiments use alternative loss functions, optimizers. learning rates, shuffle, and random parameter initializations for the training stage. Figure 4 shows the comparisons of different settings. Besides, the best way to evaluating layers of neural networks is using deconvolution to generate each layer's image [44] in image classification. In our case, although the same class has a similar FDM arrangement as shown in Figure 3, the connections between the features are undefined. This means that we cannot have a definite FDM

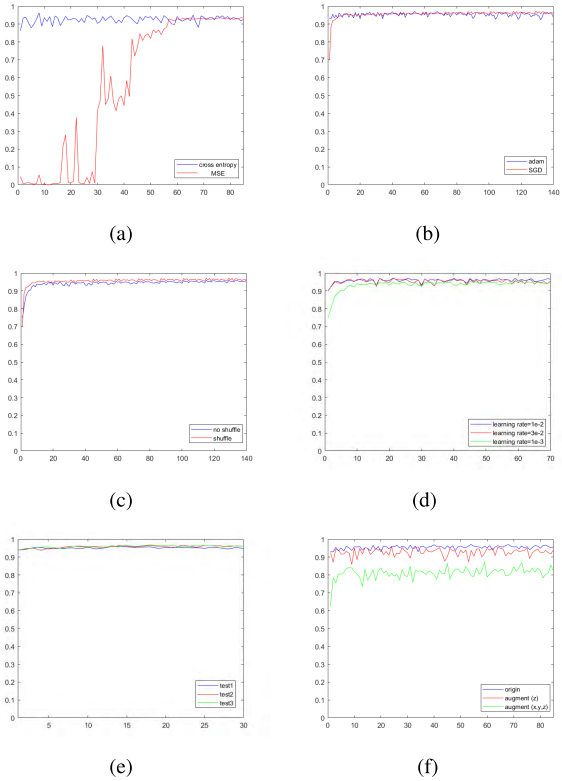

FIGURE 4. Comparisons of different settings. (a)The convergence curves of Mean Square Error (MSE) and cross entropy functions. (b)Optimizer Stochastic Gradient Descent (SGD) is a little better than Adaptive Moment Estimation(Adam).(c)Shuffle makes the accuracy better. (d)The accuracy curves for different learning rate on SGD. (e)The accuracy curves for different initial parameters.(f)The accuracy curves for data augmentation.

Algorithm 1 Our Point Cloud Classification Algorithm

1: **Training**:
2: Input the training point cloud data, i.e., the position coordinates of all points P_i, and the corresponding class labels.
3: Extract the features of each point P_i, including L_λ, V_3, P_λ, S_λ, V_1, O_λ, A_λ,E_λ,T_λ,C_λ, D_3, and Q.
4: Construct the *Feature Description Matrix* for each P_i with 16×16 size.
5: Train a deep neural network based on all the *FDM* and corresponding labels and record the neural network.
6: **Testing**:
7: Input the testing point cloud data, i.e., the position coordinates of all points P_i
8: Extract the features of each point P_i, including L_λ, V_3, P_λ, S_λ, V_1, O_λ, A_λ,E_λ,T_λ,C_λ, D_3, and Q.
9: Construct the *Feature Description Matrix* for each P_i with 16×16 size.
10: Label each point according to deep neural network based on the testing *FDM*s.

TABLE 1. The statistics of the 3D Oakland 3D Point cloud dataset.

Data	Training Data	Testing Data
Vegetation	14441	267325
Wire	2571	3794
Pole	1086	7933
Ground	4713	111112
Facade	14121	934146
All	36932	1324310

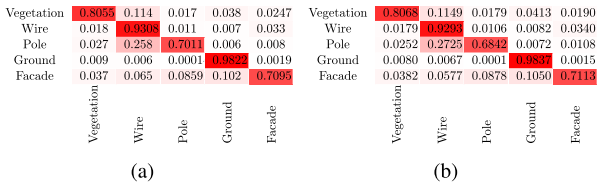

FIGURE 5. Confuse matrix comparison for the testing data. (a) The result of [3]. (b) The result of ours. Note that we normalize all the values for better observation. The point numbers are shown in the testing data column in Table 1.

for a special class, and we cannot recognize the class from the FDM directly. During the convolution process, the meaning of each layer's output is also undefined and cannot be evaluated visually. Based on the above analysis and experiments, we provide some guidelines for our training.

Guidelines for the training:

1) The parameters for the neural networks can be initialized randomly.
2) The loss function should be cross entropy function better than Mean Square Error (MSE) function, as MSE has a higher oscillation during the iterations.
3) The optimizer should be Stochastic Gradient Descent (SGD) better than Adaptive Moment Estimation (Adam).
4) The learning rate can be 0.01 (our empiric value).
5) The data should not be augmented by noises, even the quantity variances are great between different classes for the point cloud.

IV. EXPERIMENT

Our algorithm runs on a PC with Core i7 CPU 920 2.67 GHz/Intel, 3G RAM, and an NVIDIA GeForce GTX 770 video card.

A. DATASET

We test our framework on publicly 3D point cloud dataset known as Oakland 3D Point Cloud Dataset, which is one of the most widely used MLS(Mobile Laser Scanning) datasets. This dataset represents an urban environment and it is captured by a mobile platform equipped with side-looking SICK LMS laser scanners. A separation of the dataset into a training set X, a validation set V and a testing set Y are provided, and each 3D point is assigned one of the five semantic categories as Wire, Pole, Facade, Ground, and Vegetation (Figure 6(a)), with the respective number of samples per class provided in Table 1. We can extract the subtypes from them for the training and testing.

B. FDV SELECTION

The features to construct the FDV should be carefully chosen, or else the classification result maybe bad. For example, one straightforward method is to used eigenvalue-based 3D features only for classification, i.e., using $[\lambda_1, V_1, \lambda_2, V_2, \lambda_3, V_3]$ as the FDV. In this case, for the training data in Figure 6(a) and the testing data in Figure 7(a), the classification error rates are 7.5084% and 33.0057% for the training data and testing data respectively, and the 'Pole' in the testing data cannot be recognized at all. This means that using $[\lambda_1, V_1, \lambda_2, V_2, \lambda_3, V_3]$ as FDVs can classify the point cloud data to some extent, but not very practical as the accuracy is too low. If we use $[e_1, V_1, e_2, V_2, e_3, V_3]$ as the FDV, the classification error rates are 6.6717% and 30.4033% for the training and testing data respectively, and for the 'Pole' of the testing data, the accuracy is still 0. Normalization can improve the accuracy, but it ameliorates the final classification result very limited. If we use $[L_\lambda, P_\lambda, S_\lambda, O_\lambda, A_\lambda, E_\lambda, T_\lambda, C_\lambda, D_3, Q]$ as the FDV, the classification error rate of training is 10.4936%, and the classification error rate of testing is 11.6593%. Using our FDV, the classification error rate of training becomes 2.946% and the classification error rate of testing is 3.5067% with noticeably improved accuracy for the 5-categories point cloud which is reported in [3]. In [3], we use MSE function, Adam optimizer with learning rate 0.001, no shuffle, and random parameter initialization for the neural network. Based on our guideline for training, we use the cross-entropy function, SGD optimizer with learning rate 0.01 and shuffle, and also random parameter initialization for the neural network. Now the classification error rate of training and testing are decreased to 2.7835% and 3.2793% respectively. Note that we don't use the augmented data for training according to our guideline as it will impair the accuracy(see subsection IV-D). Figure 7(b) shows our testing result from a subset of the testing data, and we can see that it is very close to the ground truth.

C. CLASSIFYING DIFFERENT CATEGORIES DATA

We have tested our algorithm on data with 2 categories, 3 categories, and 4 categories, and training and testing errors are listed in Table 2, 3 and 4 respectively. In Table 2, we can see that the training errors and testing errors are quite low in most cases. Only in the case of 'Wire and Pole', the testing error is large. This is because 'Line' and 'Pole' has a similar geometric structure, making their labels cannot be effectively distinguished from each other. For other cases with more categories, both training errors and testing errors are low, which means our algorithm can classify these point clouds effectively.

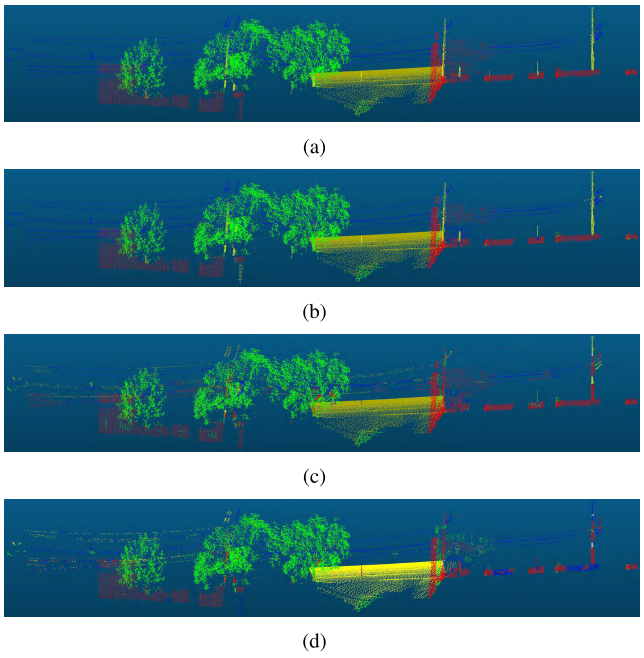

FIGURE 6. The training data and training result. (a) Point cloud with different colors corresponding different labels for training. Here red denotes for 'Facade', blue for 'Wire', grey for 'Ground', green for 'Vegetation', and yellow for 'Pole'. (b) Training result with accuracy 97.23%. Most points can be classified correctly while some points are wrongly classified such as those points in black circles. The reason for that is these points has the similar local features and even cannot be classified by human directly without the whole scene information.(c)Training result based on the augmentation data using augmentation method 1 with only 89.24% accuracy;(d)Training result based on the augmentation data using augmentation method 2 with 90.52% accuracy. This means that augment the training data will not increase the classification accuracy.

TABLE 2. Classification tests based on 2 categories.

Training Objects	Training Error	Testing Error
Vegetation and Wire	1.14%	2.47%
Vegetation and Pole	1.02%	2.59%
Vegetation and Ground	0.24%	0.63%
Vegetation and Facade	1.59%	5.83%
Wire and Pole	3.61%	47.24%
Wire and Ground	0.54%	0.30%
Wire and Facade	4.94%	4.57%
Pole and Ground	0.05%	0.15%
Pole and Facade	1.52%	1.73%
Ground and Facade	0.06%	5.06%

D. DATA AUGMENTATION

As different classes have different point numbers, is it necessary to expand those classes with small point numbers for training? As shown in Table 1, the numbers for Wire, Pole and Ground in the training data are too much less than the numbers of Vegetation and Facade. We can augment these three classes database on [36] using two methods: (1) adding the Gaussian white noise with a mean of 0 and a variance of 0.02 to each point's three-dimensional coordinates (x, y, z); (2) Only the z-axis coordinate of each point is added with Gaussian white noise with a mean of 0 and a variance of 0.02. Specifically, we augment Wire 6 times, Pole 14 times, and Ground 3 times, making the number of Vegetation, Wire, Pole, Ground and Facade be 14441,15426,15206,14139, and 14121 respectively. However, both data augmentation methods impair the classification accuracies as shown in Figure 6(c), Figure 6(d) and Figure 4(f). The main reason is that these augmented points can be seen as noise points and will disturb the feature extraction, making the FDM unable to reflect the

TABLE 3. Classification tests based on 3 semantic categories.

Training Objects	Training Error	Testing Error
Vegetation, Wire and Pole	2.64%	6.42%
Vegetation, Wire and Ground	0.94%	1.09%
Vegetation, Wire and Facade	3.31%	7.03%
Vegetation, Pole and Ground	1.00%	1.19%
Vegetation, Pole and Facade	2.39%	6.20%
Vegetation, Ground and Facade	1.27%	1.80%
Wire,Pole and Ground	1.14%	0.568%
Wire, Pole and Facade	5.48%	5.76%
Wire, Ground and Facade	1.70%	10.52%
Pole, Ground and Facade	0.91%	1.06%

TABLE 4. Classification tests based on 4 semantic categories.

Training and Testing Objects	Training Error	Testing Error
Vegetation, Wire, Pole and Ground	1.78%	2.16%
Vegetation, Wire, Pole and Facade	4.80%	10.83%
Vegetation, Wire,Ground and Facade	2.78%	1.87%
Vegetation, Pole Ground and Facade	1.80%	2.47%
Wire, Pole, Ground and Facade	2.32%	2.79%

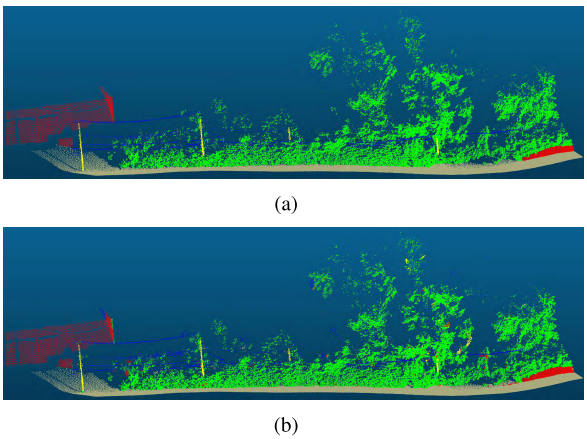

FIGURE 7. The ground truth and the testing result from the testing data. (a) Groundtruth labels for the subset of the testing data, and the color meaning is as the same as in Figure 6(a). (b) The classification result using our FDM and our trained CNN from Figure 6.

TABLE 5. The classification accuracy based on different classifiers(%). The bold numbers are the highest accuracies among all the methods.

Data	NN	DT	NB	LDA	QDA	SVM	RF	RFe	AB	MLP	[3]	Ours
Vegetation	75.38	59.56	**83.09**	80.02	82.49	79.46	81.41	82.79	82.74	77.26	80.55	80.68
Wire	80.13	73.48	79.15	87.00	79.13	82.99	86.05	82.01	80.40	81.11	**93.08**	92.93
Pole	74.49	71.36	76.33	79.85	76.28	78.10	79.99	73.25	70.17	**82.07**	70.11	68.42
Ground	84.06	83.90	90.47	96.24	90.70	94.92	**98.48**	96.58	98.41	92.70	98.22	98.37
Facade	55.82	46.63	51.88	67.12	52.15	64.39	67.01	58.87	65.73	66.43	70.95	**71.13**
All	79.87	75.76	85.63	90.39	85.69	89.10	92.25	90.45	92.28	87.29	94.68	**94.75**

local features. This means we should use the original data for feature extraction but not augment the data just for balancing the number of samples.

E. PERFORMANCE ANALYSIS

The main cost of our algorithm is on the feature extraction. For a point cloud with N points, we use kd-tree to implement the KNN algorithm, and the average computation complexity is $O(NlogN)$. As we choose constant k-nearest neighbors to construct FDM for training and testing, the time will also be $O(NlogN)$ for testing after the neural network is obtained. For the training point cloud with 36932 points, the feature extraction will spend 19.95 seconds, and the whole training process will spend a few minutes. For the testing stage, the feature extraction will also spend a few seconds, and then the prediction will be finished immediately based on our CNN structure.

F. COMPARISON

Based on the optimal neighborhood, Weinmann *et al.* [12] employ different classifiers for the point cloud classification on the same data set, and we make a comparison between our methods with them as shown in Table 5. In the table, NN denotes Nearest Neighbor classifier [13], DT for decision trees [14], NB for Naive Bayesian [15], LDA for Linear Discriminant Analysis, QDA for Quadratic Discriminant Analysis, SVM for Support Vector Machines [16], RF for Random Forests [45], RFe for Random Fern (RFe) classifier [17], AB for Adaptive Boosting [46], and MLP for Multi-Layer Perceptron [18]. From Table 5, we can see that our method have the best overall accuracy, and the classification accuracies are also high compared to these methods based on the optimal neighborhood. In the case of point cloud data in a large scene with only three-dimensional coordinate information, we construct the Feature Description Matrix(FDM) of each point by concatenate the Feature Description Vectors(FDV) composed of the extracted features of each point according to the nearest neighbor principle as the input of the neural networks. The FDM is independent with the order of points, so that the neural network can learn the features further. It enhances the 3D point cloud auto analysis and accuracy classification in the Large-scale scene. Especially that our method tries to classify each point, while Pointnet [36] or Pointnet++ [37] method mainly try to classify a set of points. The point clouds from [36] are mainly sampled from the surface of the model. These methods mainly for 3D shape classification but not for the large scene point cloud classification. Compared to [3], the classification accuracy is improved from 97.05% to 97.23% for the training data in Figure 6(a) and from 94.68% to 94.75% for the testing data.

V. CONCLUSION AND FUTURE WORK

A classification algorithm for large-scale 3D point cloud scene combined Feature Description Matrices(FDM) is given. Our method integrates traditional feature extraction process and the deep learning thereby obtains a higher classification accuracy. We employ our Feature Description Matrix that is suitable for convolution neural networks and achieve a higher classification accuracy. We also give guidelines for the training stage.

In the future, more different deep learning models should be tested in our algorithm, and the principle of the arrangement for the FDV should be studied further, with the effectiveness of mesh models should be tested. We should also accelerate the feature extraction process in order to classify the point cloud faster.

REFERENCES

[1] J. M. Nield, F. S. G. Wiggs, and R. S. Squirrell, "Aeolian sand strip mobility and protodune development on a drying beach: Examining surface moisture and surface roughness patterns measured by terrestrial laser scanning," *Earth Surf. Processes Landforms*, vol. 36, no. 4, pp. 513–522, Apr. 2015.

[2] W. Hao and Y. Wang, "Classification-based scene modeling for urban point clouds," *Opt. Eng.*, vol. 53, no. 3, Mar. 2014, Art. no. 033110.

[3] L. Wang, W. Meng, R. Xi, Y. Zhang, L. Lu, and X. Zhang, "Large-scale 3D point cloud classification based on feature description matrix by CNN," in *Proc. 31th Int. Conf. Comput. Animation Social Agents*, May 2018, pp. 43–47.

[4] Z. Zhang *et al.*, "A multilevel point-cluster-based discriminative feature for ALS point cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3309–3321, Jun. 2016.

[5] Z. Wang *et al.*, "A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2409–2425, May 2015.

[6] B. Yang, Z. Dong, Y. Liu, F. Liang, and Y. Wang, "Computing multiple aggregation levels and contextual features for road facilities recognition using mobile laser scanning data," *ISPRS J. Photogram. Remote Sens.*, vol. 126, pp. 180–194, Apr. 2017.

[7] A. Martinovic, J. Knopp, H. Riemenschneider, and L. D. Van Gool, "3D all the way: Semantic segmentation of urban scenes from start to end in 3D," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4456–4465.

[8] N. Chehata, L. Guo, and C. Mallet, "Airborne lidar feature selection for urban classification using random forests," *Int. Archives Photogram., Remote Sens. Spatial Inf. Sci.*, vol. 38, no. 1, p. W8, 2009.
[9] B. Guo, X. Huang, F. Zhang, and G. Sohn, "Classification of airborne laser scanning data using JointBoost," *ISPRS J. Photogramm. Remote Sens.*, vol. 100, pp. 71–83, Feb. 2015.
[10] M. Kragh, R. N. Jørgensen, and H. Pedersen. "Object detection and terrain classification in agricultural fields using 3d lidar data," in *Computer Vision Systems*, L. Nalpantidis, V. Krüger, J. O. Eklundh, and A. Gasteratos, Ed., Cham, Switzerland: Springer, 2015, pp. 188–197.
[11] N. Brodu and D. Lague, "3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology," *ISPRS J. Photogram. Remote Sens.*, vol. 68, pp. 121–134, Mar. 2012.
[12] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS J. Photogram. Remote Sens.*, vol. 105, pp. 286–304, Jul. 2015.
[13] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
[14] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[15] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proc. 7th Conf. Uncertainty Artif. Intell.*, Aug. 1995, pp. 338–345.
[16] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[17] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[18] M. A. Riedmiller and H. A. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. On Neural Netw.*, vol. 1, Apr. 1993, pp. 586–591.
[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[20] W. Shao and S. Du, "Spectral-spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Oct. 2016.
[21] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proc. CVPR*, Jun. 2016, pp. 5648–5656.
[22] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural networks by a generalized discriminant analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 596–608, Apr. 2012.
[23] S. Serikawa and H. Lu, "Underwater image dehazing using joint trilateral filter," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 41–45, Jan. 2014.
[24] H. Lu, Y. Li, S. Mu, D. Wang, H. Kim, and S. Serikawa, "Motor anomaly detection for unmanned aerial vehicles using reinforcement learning," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2315–2322, Aug. 2018.
[25] H. Lu *et al.*, "Conet: A cognitive ocean network," *IEEE Wireless Commun.*, to be published.
[26] H. Lu, Y. Li, T. Uemura, H. Kim, and S. Serikawa, "Low illumination underwater light field images reconstruction using deep convolutional neural networks," *Future Gener. Comput. Syst.*, vol. 82, pp. 142–148, May 2018.
[27] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain intelligence: Go beyond artificial intelligence," *Mobile Netw. Appl.*, vol. 23, no. 2, pp. 368–375, Apr. 2018.
[28] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.
[29] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the Art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.
[30] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Proc. NIPS*, 2011, pp. 1–9.
[31] Z. Wu *et al.*, "3d shapenets: A deep representation for volumetric shapes," in *Proc. CVPR*, May 2015, pp. 1912–1920.
[32] J. Huang and S. You, "Point cloud labeling using 3D convolutional neural network," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2016, pp. 2670–2675.
[33] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jul. 2017, pp. 1–9.
[34] F. Liu *et al.*, "3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds," in *Proc. ICCV*, Oct. 2017, pp. 5679–5688.
[35] A. Boulch, B. L. Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Proc. Eurographics Workshop 3D Object Retr.*, 2017, pp. 1254–1268.
[36] C. R. Qi, H. Su, K. Mo, and J. L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. CVPR*, Jul. 2017, pp. 77–85.
[37] C. R. Qi, L. Yi, H. Su, and J. L. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. NIPS*, 2017, pp. 5099–5108.
[38] K. Guo, D. Zou, and X. Chen, "3D mesh labeling via deep convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 1, pp. 3:1–3:12, 2015.
[39] L. Luciano and A. ben Hamza, "Deep learning with geodesic moments for 3D shape classification," *Pattern Recognit. Lett.*, vol. 105, pp. 182–190, Apr. 2017.
[40] M. Weinmann, B. Jutzi, and C. Mallet, "Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features," *ISPRS Ann. Photogram., Remote Sens. Spatial Inf. Sci.*, vol. 3, pp. 181–188, Sep. 2014.
[41] M. Pauly, M. Gross, and L. P. Kobbeltm, "Efficient simplification of point-sampled surfaces," in *Proc. IEEE Visualizat.*, Nov. 2002, pp. 163–170.
[42] M. Weinmann, B. Jutzi, and C. Mallet, "Feature relevance assessment for the semantic interpretation of 3d point cloud data," *ISPRS Ann. Photogram., Remote Sens. Spatial Inf. Sci.*, vol. 5, pp. 313–318, Nov. 2013.
[43] X. Liu, H. Li, W. Meng, S. Xiang, and X. Zhang, "3d point cloud classification based on discrete conditional random field," in in *Proc. Int. Conf. Technol. E-Learn. Digital Entertainment*, 2017, pp. 115–137.
[44] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[45] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[46] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

LEI WANG received the M.Sc. degree in earth exploration and information processing from the East China University of Technology. She is currently pursuing the Ph.D. degree with the School of Computer, Northwestern Polytechnic University. She is an Associate Professor with the School of Information Engineering, East China University of Technology. Her current research interests include computer vision, 3D-data analysis, and deep learning.

WEILIANG MENG received the B.S. degree in computer science from the Civil Aviation University of China, in 2003, the M.Sc. degree in computer application from Tianjin University, in 2006, and the Ph.D. degree in computer application from the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, in 2010, where he is currently an Associate Professor with the National Laboratory of Pattern Recognition, Institute of Automation. His research interests include point cloud processing, computer graphics, computer vision, and deep learning.

RUNPING XI received the Ph.D. degree in computer science and technology from Northwestern Polytechnical University, in 2011, where he is currently an Associate Professor with the School of Computer. His current research interests include computer graphics, image processing, computer vision, and multimedia information processing.

YANNING ZHANG received the Ph.D. degree in computer science and technology from Northwestern Polytechnical University, where she is currently a Professor with the School of Computer. Her research interests include image processing, computer vision, pattern recognition, artificial intelligence, and machine learning.

CHENGCHENG MA received the B.S. degree from the Honor College, Northwestern Polytechnic University, in 2017. He is currently pursuing the master's degree with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image processing and deep learning.

LING LU received the B.S. degree in geophysical exploration from the East China University of Technology, where she is currently a Professor with the School of Information Engineering. Her research interests include computer graphics, digital image processing, and point cloud reconstruction.

XIAOPENG ZHANG received the Ph.D. degree in computer application from the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, where he is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation. His main research interests include computer graphics, computer vision, and artificial intelligence.