Classification of 3D Point Clouds By A New Augmentation Convolutional Neural Network

Sheng Xu, Xuan Zhou, Weidu Ye, Qiaolin Ye*

***Abstract*—Nowadays, the classification of point clouds has become a fundamental problem in 3D information study. Different from the deep learning process of natural images, 3D point clouds are massive and unorganized, which can be difficultly captured features by the convolution process directly. This letter proposes a new augmentation convolutional neural network (ACNN) to classify point clouds by adding a key augmentation layer before the classical sampling and convolution structure. Input data will be augmented before each sampling layer, which brings abundant learning information to help the network capture more local structures. In order to make the augmentation more effective, we formulate parameters of augmentation layers learnable in the learning process according to the loss function. The proposed augmentation is based on automatically tuning the magnitude of the smoothness, which plays a significant role in point cloud processing and provides local features, e.g. edges, contours, and edges. Results show that we have achieved the overall accuracy of 92.52% and 89.11% in the object classification on ModelNet10 and ModelNet40, respectively, which shows our superiority over other methods. Besides, the ACNN achieves an average miscalculation error of 0.28 and cross-entropy loss of 0.48 in the classification of laser scanning point clouds, which shows high robustness to noise and density in the outdoor scene classification.**

***Index Terms*—Deep learning, CNNs, classification, point clouds, augmentation layer**

I. Introduction

A fundamental work in the 3D point cloud study is the object classification, which plays an important role in the geometric analysis [1], vegetation delineation [2] and segmentation [3]. Object classification is the process of assigning a unique label to points from one kind of instance. In general, the classification methods are either unsupervised or supervised. The former focuses on the feature proximity [4], which minimizes the feature distance between objects with the same label. The latter tries to fit a function for splitting objects with different labels, which is achieved by a learning process. Since the 3D feature calculation in point clouds is still a challenging task, this work puts efforts into the machine learning methods to classify objects. Different from the classical classification approaches based on the handcraft features [5], we focus on deep learning models.

Commonly used deep learning methods related to the point cloud classification follow a sampling and convolution process, which are present as convolutional neural networks (CNNs). In the method of [6], the authors propose a deep hierarchical feature learning method to capture local patterns, and improve their generalization to various point densities by a sampling strategy. However, the formulation of multi-scale features arises much calculation in local regions. In the method of [7], they present a VoxelNet to recognize objects using a 3D bounding box. They formulate a new encoding layer to obtain features from local voxels, and then integrate local features to obtain 3D shape information. Although the VoxelNet works well in binary classification, it may fail in multiply labels classification when input data are limited and sparse. In the method of [8], the authors develop a permutation invariant convolution to improve training speed in CNNs at a less complex network architecture. They define representative features and resolve the point order ambiguity to help CNNs work with larger receptive fields. Although their algorithm works well in both indoor and outdoor scenes, it is not easy to formulate a robust operator for different environments. In the method of [9], the authors propose 3D CNNs to analyze sparse point clouds. Although their algorithm implements the convolution process directly on the collection of shallow octrees, a conditional random field model is necessary to be utilized for learning global features.

Since point clouds are unorganized and massive, it is difficult to capture enough features directly from 3D point clouds. Although the existing CNNs have achieved promising results in the point cloud classification, there are three remaining issues. First, the learning task in 3D point clouds requires a large size of memory and high computation. Second, point clouds are uneven which makes the feature extraction difficult. Third, the learning samples for the training process are usually limited.

Different from the existing CNNs for the point classification, we develop an augmentation layer to improve our generalization in the learning process, which helps neural networks capture point cloud features in 2D projection space abundantly and accurately. It is worth noting that the projected points can not be directly processed in CNNs as other natural images do. The difference between the point image and natural image lies in that a natural image is present by topologically organized pixels. However, a point image is created by the coordinate projection, whose elements are still unorganized and massive. Hence, our contributions include 1) the formulation of a new augmentation layer for CNNs to capture local features of input data, and 2) the establishment of a learnable Gaussian filtering process for the classification of massive and uneven points' images, which helps us reduce the memory and computation.

Corresponding author: Qiaolin Ye (yqlcom@njfu.edu.cn)

Sheng Xu, Xuan Zhou and Qiaolin Ye are with the College of Information Science and Technology, Nanjing Forestry University, Nanjing 210037, China.

Weidu Ye is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.

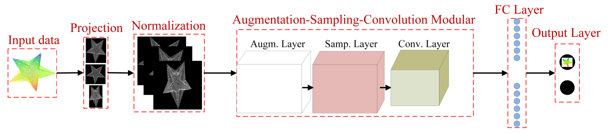

Fig. 1. Structure of the augmentation convolutional neural network.

II. The Augmentation Convolutional Neural Network

This section aims to develop a new augmentation layer for the classical convolutional neural networks. Before we present the augmentation convolutional neural network (ACNN), it is necessary to discuss the data preprocessing for the input layer. Our input data are point clouds, i.e. a set of points with the coordinate (x, y, z), which are different from 2D images at three aspects. 1) There is only coordinate information in raw point clouds, which makes the convolution process difficult to understand. (2) The input samples are limited and insufficient for the 3D training process because the split of 3D point clouds is very complicated and time-consuming. (3) Input point clouds are uneven and unorganized, which are quite different from pixels. In the preprocessing, we first project point clouds into 2D space as shown in Fig.1. Then, we use the zero-padding strategy to make input data as a square image at a fixed size. After those steps, we obtain the normalized data as input for the subsequent augmentation-sampling-convolution modular and full connection layer. The output layer demonstrates the results of the classification.

A. Network formulation

The augmentation layer aims to capture features from input data at different views, which releases dimension loss and the sample limitation. Furthermore, we smooth input 2D images in this layer to achieve features from scattering points. We develop the augmentation layer before the sampling layer as shown in Fig.2. The input data of augmentation layers are the projection images of 3D point clouds at different views, e.g. along the axis, and the output data of those layers are different smoothed images.

Denote the augmentation layer as A_i, the sampling layer as S_i, and the convolution layer as C_i, where i represents the index of layers as demonstrated in Fig.2. To describe the network structure, we use $I_i(x, y, m)$ to display the size of data in each layer, where x and y are the size of each feature map and m shows the number of kernels in the current layer. We use a_i and c_i to show the size of kernels in the current augmentation layer and convolutional layer, respectively. The coefficient r_i shows the sampling ratio in the current sampling layer. The number of kernels in A_i depends on the number of kernels in its prior layer and the size of its kernel decided by the subsequent S_{i+1} and C_{i+2}, which means that a_i is $(r_{i+1} \cdot c_{i+2})$.

Input images are smoothed in the augmentation layer, and then data will be put forward to sampling and convolution layers. We use an example to demonstrate the structure of the ACNN and its forward propagation in the learning process. Let

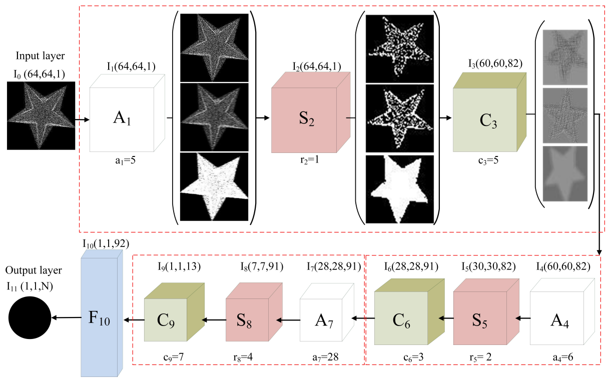

Fig. 2. Network structure of the augmentation-sampling-convolution modular.

the input image as $I_0(64, 64, 1)$. Based on the r_2 and c_3 set by the user, the kernel in A_1 is $a_1 = 5$. For the boundary regions we choose the zero-padding strategy, hence, there is no size reduction in the augmentation layer, i.e. we have $I_1(64, 64, 1)$. The size of feature maps in S_2 and C_3 depends on r_2 and c_3, which is obtained by the basic propagation rules in CNNs. Repeat our modular to generate layers from A_4 to C_9 as shown in Fig.2. Then a full connection layer is chosen to capture global features from C_9 to obtain F_{10}. Finally, the output layer $I_{11}(1, 1, N)$ shows results, where N stands for the number of output nodes.

B. Parameters update

The proposed ACNN has three learnable parameters to be updated in the learning process, namely the weight and bias in CNN models and a new augmentation **U**. **U** aims to augment input data by using Gaussian function to smooth points for achieving neighbor information. The propagation function in the augmentation layer is

$$\mathbf{Y} = \frac{1}{2}(\mathbf{I} - \mathbf{U})^2 \cdot \mathbf{X} + \mathbf{G} * \mathbf{X}, \ \ \mathbf{U} = \{u_1, u_2, ..., u_k, ...\}, \quad (1)$$

where **X** and **Y** are input data and output data of this layer, respectively. **U** is a vector to describe the augmentation for each feature map, which is similar to the bias. Each map shares with only one u_k. **I** is a vector at the same dimension with **U**. '$*$' shows the convolution process between **G** and **X**, where the element (i, j) of the matrix **G** is defined as

$$G_{i,j} = \exp\left(-\frac{(x_i - x_0)^2 + (y_i - y_0)^2}{2\sigma^2}\right) \cdot \lambda_{i,j}, \quad (2)$$

$$\lambda_{i,j} = \begin{cases} u_k & , i \neq j; \\ 1 & , i = j. \end{cases} \quad (3)$$

Coefficients i and j depend on the kernel size of a_i and range from $i_o - a_i$ to $i_o + a_i$ and $j_o - a_i$ to $j_o + a_i$, respectively, where i_o and j_o are the coordinate of the current neuron. The function of the augmentation layer can be regarded as a special multi-scale Gaussian filter. Rather than tuning the value of sigma in Eq.(1), we optimize the intensity value of neighbors in the learning processing, which is robust to the density changing.

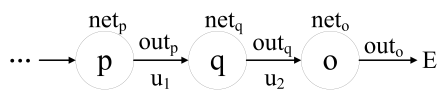

Fig. 3. Parameter propagation of $\mathbf{U} = \{u_1, u_2\}$.

Let us focus on Eq.(1). When u_k is close to 0, the effect of the smoothness will vanish. When increases u_k to 1, the effect of the original data tends to be ignored. The value of u_k tunes the smoothness of input data based on Gaussian function as shown in Eq.(2). Let's keep increasing u_k to a large number ($\gg 1$), intensities of input data will be over-exposed and they grow to 255. When it turns to 255, input data will be regarded as binary images. The classification will purely depend on those binary images. If it turns to 0, input data will turn black, which will be discarded in the network processing.

The loss function is usually based on the cross-entropy and the update of the weight and bias uses the backpropagation technique. Similarly, we can also update $\mathbf{U}$ in the same way based on the loss error E as shown in Eq.(4), where η is a user-defined learning rate. To make it easy to follow, we ignore the sampling and convolution layer in the propagation as shown in Fig.3.

$$u_k^+ = u_k - \eta \cdot \frac{\partial E}{\partial u_k}. \tag{4}$$

As shown in Fig.3, p and q are two hidden nodes and o is the output node. net_p and out_p are the input and output of p, respectively. The same to the node q and o. We have $out_i = f(net_i)$, where the activation function f can be ReLU or LeakyReLU and we denote $f' = \frac{\partial out_i}{\partial net_i}$. Based on Eq.(1), we have

$$\begin{aligned} net_o(i,j) =& \frac{1}{2}(1-u_2)^2 \cdot out_q(i,j) \\ &+ \sum_{i-a_o}^{i+a_o} \sum_{j-a_o}^{j+a_o} \left(G(i,j) \cdot out_q(i,j)\right). \end{aligned} \tag{5}$$

Denote E' as the derivative of E with respective to the output of layers and let G' as the derivative of G with respective to u_k. We have

$$\begin{aligned} \frac{\partial E}{\partial u_2} = \frac{\partial E}{\partial out_o} \cdot \frac{\partial out_o}{\partial net_o} \cdot \frac{\partial net_o}{\partial u_2} = E' \cdot f' \cdot (u_2 - 1) \cdot out_q(i,j) \\ + \sum_{i-a_o}^{i+a_o} \sum_{j-a_o}^{j+a_o} \left(G'(i,j) \cdot out_q(i,j)\right). \end{aligned} \tag{6}$$

Therefore,

$$\begin{aligned} \frac{\partial E}{\partial u_1} &= \frac{\partial E}{\partial net_q} \cdot \frac{\partial net_q}{\partial u_1} \\ &= \frac{\partial E}{\partial out_o} \cdot \frac{\partial out_o}{\partial net_o} \cdot \frac{\partial net_o}{\partial out_q} \cdot \frac{\partial out_q}{\partial net_q} \cdot \frac{\partial net_q}{\partial u_1}. \end{aligned} \tag{7}$$

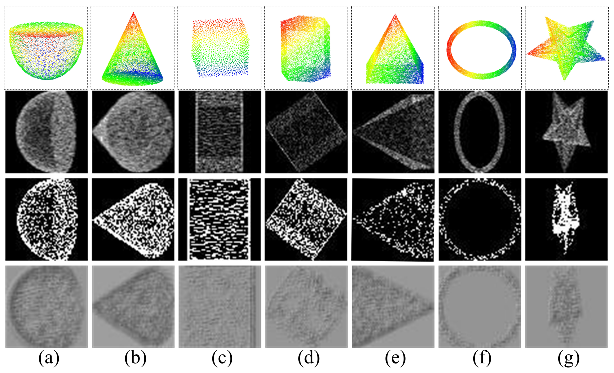

Fig. 4. Processing of synthetic point cloud sets. (a) Hemispheres. (b) Cones. (c) Cubes. (d) Cylinders. (e) Pyramids. (f) Rings. (g) Stars.

TABLE I
ACCURACY OF DIFFERENT ACNN STRUCTURES.

ID	NoL	ASC	r	c	F	Size	Loss	MCR
1	7	2	(1,4)	(5,7)	1	32	0.06	0.13
2	10	3	(1,2,4)	(5,3,3)	1	32	0.02	0.01
3	13	4	(1,1,2,4)	(3,3,3,3)	1	32	0.11	0.29
4	11	3	(1,2,4)	(5,3,3)	2	32	0.25	0.34
5	12	3	(1,2,4)	(5,3,3)	3	32	0.36	0.51
6	10	3	(1,2,4)	(5,3,7)	1	64	0.18	0.23

III. EXPERIMENTS AND EVALUATIONS

A. Performance on the 3D object classification

We apply the proposed ACNN on a synthetic point cloud dataset containing seven 3D polyhedron objects generated by Sketchup (www.sketchup.com), namely hemispheres, cones, cubes, cylinders, pyramids, rings, and stars as shown in Fig.4. In this case, each polyhedron set contains 100 instances, and we set the ratio of the training set, validation set, and test set for each label as 7:1:2, respectively. The training set is used to tune weights for the network, the validation set is prepared for the detection of the overfitting, and the test set is designed for the accuracy calculation.

In preprocessing, the projection is based on the rotation of 3D point clouds along the axis. The evaluation metrics include cross-entropy loss error (Loss) and miscalculation error (MCR). We try different ACNN structures and show results in Table I with the same number of kernels in each layer. The first column 'ID' means the experiment number, 'NoL' means the number of layers in the network, 'ASC' means the number of augmentation-sampling-convolution modular, 'r' means the ratio of each sampling layer, 'c' means the kernel size of each convolution layer, 'F' means the number of full connection layer, 'Size' is the size of input data, 'Loss' is the entropy loss error and 'MCR' is the miscalculation error calculated by the ratio of correctly classified objects.

As shown in Table I, results highly depend on the network structure, and we achieve the best performance at ID #2. We use 13,22,17,36 for the kernels of the first, second, third convolution layer and the full convolution layer to generate feature maps. In this case, our ACNN works well at 10 layers. When we enlarge the size of input data or increase the number of layers, the accuracy did not show any improvements, because

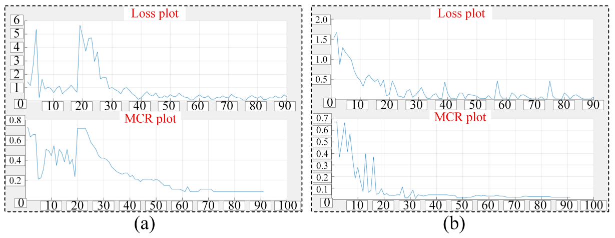

Fig. 5. Accuracy plot of the ACNN. (a) Results of the ACNN on synthetic point clouds without augmentation layers. (b) Results of the ACNN on synthetic point clouds with augmentation layers.

we incur more parameters in the training, which makes $\mathbf{U}$ be a high-dimensional vector. Because of the complicated error propagation, we suggest users stay at a less complex network structure to keep a low dimensional vector in $\mathbf{U}$, otherwise, users are required to choose better optimization methods to address the overfitting. The size of the input data is determined empirically. Experiments show that in the case of basic geometrical shape object classification, ACNN does not ask for a large resolution, when capturing the local features. Data with a large resolution requires more complex network layers in the optimization, otherwise, the accuracy can be undesired. Results of the augmentation, sampling, and convolution in the first ASC are shown in the second, third, and fourth row of Fig.4, respectively. In the evaluation, we use both LOSS and MCR metrics. The dissimilarity lies in that LOSS focuses on the evaluation of the optimization error and MCR intends to measure the object classification. Although the LOSS can be periodic rise and fall, the classification error can still be converged which is based on the activation function at the end of the networks.

For the ablation analysis, we show results of ordinary convolution layers with nonlinear activation functions. We fix the $\mathbf{U}$ as a constant 0, which means that we turn off the augmentation layer. At this time, both the loss error and MCR increase greatly as shown in Fig.5(a) compared with results of ID #2 as shown in Fig.5(b). Experiments show that the proposed augmentation layer plays a significant role in capturing local features and we have achieved a higher improvement than classical CNNs. Please be aware that our work does not require projection plane optimization because we have used redundant projection planes in the network processing. Although it may increase the time cost, the proposed ACNN succeeds in providing high-quality results by checking information from all provided projection data.

For comparison of our methods with others, we test the ACNN on the public object classification benchmark ModelNet [10] containing a comprehensive clean collection of 3D CAD models for objects. ModelNet10 and ModelNet40 contain CAD models from the 10 and 40 categories, respectively. The training and testing split is included in the file. The CAD models are completely cleaned in-house, and the orientations of the models are manually aligned by the developers.

Our comparison results are shown in Table II. The displayed methods contain VoxelNet [11], GIFT [12], Zanuttigh and Minto (ZM) [13], binVoxNetPlus (bVNP) [14] and Primitive-GAN (PG) [15]. Accuracy values are obtained from the online benchmark (http://modelnet.cs.princeton.edu, accessed in 2021/11/30).

TABLE II
COMPARISON WITH THE OTHER APPROACHES (%).

Accuracy	VN	GIFT	ZM	bVNP	PG	ACNN
ModelNet10	92.00	92.35	91.50	92.32	92.20	92.52
ModelNet40	83.00	83.10	87.80	85.47	86.40	89.11

In the comparison, VN is based on the binary occupancy grid information, which provides an easy 3D convolution process in ModelNet classification. Similarly, bVNP provides binary volumetric CNNs for 3D object recognition, which transforms the inputs and weights to binary values. GIFT uses their 3D shape search engine based on the projective images of 3D shapes, which provides efficient results in the classification when input data are collected with a similar density. The accuracy of these three methods degrades greatly when adding more labels in the classification. ZM takes a multi-branch convolutional neural network for shape classification. PG uses a factorized generative model for 3D shape generation, which chooses the primitive parts for shapes as attributes. Those two methods achieve high accuracy on both ModelNet10 and ModelNet40, but ZM requires an extra linear classifier for the vector classification, and PG requires the multi-view rendering process to capture more features. Our ACNN is in an end-to-end framework, which captures local features through the proposed augmentation layers. Although our method fails to be better than others in some CAD models, such as the bench, sofa, and table, we show a high performance in the overall accuracy than others.

B. Performance on the laser scanning point classification

As shown above, the proposed method works well in 3D object classification. In order to show our robustness, we also test the performance on the laser scanning point classification, i.e. semantic segmentation. One key difference between object classification and semantic segmentation is the input data. The former input requires input 3D instance, and the latter asks for original points which can be difficult because of numerous noise points and points' various densities.

In order to apply our work to the classification of outdoor laser scanning point clouds, we first conduct the instance clustering on the input scene. If the input scene is very complex, users can split it into small scales. Each instance is prepared as an input for the ACNN. Experimental data are collected by Riegl VMX-450. Laser data collection density was calculated by selecting 1 m radius patches along the vehicle trajectory every 50 m. A mean point density along the trajectory of approximately 750 pts/m^2. All areas to be scanned met a minimum threshold of 500 pts/m^2 along the hard surfaces. The learning process includes 125 bins, 428 buildings, 399 low vegetation, 167 pedestrian, 163 poles, 88 traffic signs, 596 high vegetation, and 460 vehicles as shown in Fig.6.

In this case, our LOSS and MCR are 0.48 and 0.28, respectively. Results of the augmentation, sampling, and convolution

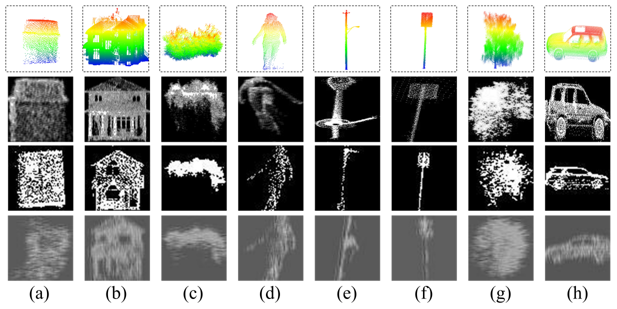

Fig. 6. Processing of laser scanning point clouds. (a) Bins. (b) Buildings. (c) Low vegetation. (d) Pedestrian. (e) Poles. (f) Traffic signs. (g) High vegetation. (h) Vehicles.

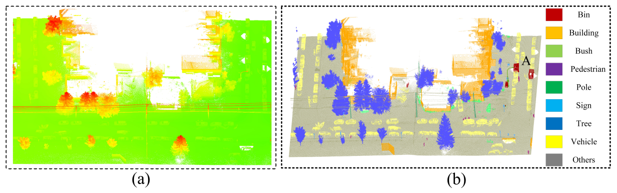

Fig. 7. Qualitative analysis. (a) Input laser scanning point clouds. (b) Semantic segmentation results by the ACNN.

from the first ASC are shown in the second, third, and fourth row of Fig.6. For the qualitative analysis, we demonstrate classification results in Fig.7 showing that the ACNN works effectively in point classification, especially in planar or linear objects.

It is worth noting that in the case of semantic segmentation, we are required to perform instance segmentation first, and then transform points from each instance into a 2D space image for the ACNN. Although our transformation processing manner brings information losses, we can increase the number of projection views to help capture more features. The key addressed issue lies in the classification of unorganized point clouds. The projection of point clouds on 2D space is a set of scatter points, which can be trained in our ACNN, and there is no need to fuse the captured 2D features and the original 3D point clouds.

IV. Conclusion

In this work, we address the feature limitation in CNNs, and propose an augmentation convolutional neural network (ACNN), which is a novel trainable deep architecture for the classification of laser scanning point clouds.

The augmentation magnitude is updated and optimized in the learning process, which provides abundant feature maps for point clouds and helps the network capture more local features, e.g. edges, contour, and curves. These contributions enable us to obtain a high performance in the point classification. The proposed ACNN has achieved the overall accuracy of 92.52% and 89.11% in ModelNet10 and ModelNet40, respectively, and the misclassification error of 0.28 on laser scanning point clouds, which shows that we are effective in both the object classification and the semantic segmentation at a less complex network. In future work, it is worthwhile trying to tune the projection angle of the proposed network in the learning process automatically. It is also interesting to drop out the potential redundant features maps in the augmentation for less computation.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62102184), in part by the Natural Science Foundation of Jiangsu Province (Grant No. BK20200784), in part by the Natural Science Foundation of the Higher Education Institutions of Jiangsu Province (Grant No. 19KJB520010), and in part by China Postdoctoral Science Foundation (Grant No. 2019M661852).

References

[1] S. Xia, D. Chen, R. Wang, J. Li, and X. Zhang, "Geometric primitives in lidar point clouds: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 685–707, 2020.

[2] T. Yun, K. Jiang, G. Li, M. P. Eichhorn, J. Fan, F. Liu, B. Chen, F. An, and L. Cao, "Individual tree crown segmentation from airborne lidar data using a novel gaussian filter and energy function minimization-based approach," *Remote Sensing of Environment*, vol. 256, p. 112307, 2021.

[3] C. Hu, Z. Pan, and T. Zhong, "Leaf and wood separation of poplar seedlings combining locally convex connected patches and k-means++ clustering from terrestrial laser scanning data," *Journal of Applied Remote Sensing*, vol. 14, no. 1, p. 1, 02 2020.

[4] Y. Yu, H. Guan, D. Li, S. Jin, T. Chen, C. Wang, and J. Li, "3-d feature matching for point cloud object extraction," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 2, pp. 322–326, 2019.

[5] Q. Li, P. Yuan, Y. Lin, Y. Tong, and X. Liu, "Pointwise classification of mobile laser scanning point clouds of urban scenes using raw data," *Journal of Applied Remote Sensing*, vol. 15, no. 2, p. 024523, 2021.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[7] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[8] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1607–1616.

[9] F. Wang, Y. Zhuang, H. Gu, and H. Hu, "Octreenet: A novel sparse 3-d convolutional neural network for real-time 3-d outdoor scene analysis," *IEEE Transactions on Automation Science and Engineering*, 2019.

[10] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[12] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki, "Gift: A real-time and scalable 3d shape search engine," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5023–5032.

[13] P. Zanuttigh and L. Minto, "Deep learning for 3d shape classification from multiple depth maps," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3615–3619.

[14] C. Ma, W. An, Y. Lei, and Y. Guo, "Bv-cnns: Binary volumetric convolutional networks for 3d object recognition." in *BMVC*, vol. 1, no. 2, 2017, p. 4.

[15] S. H. Khan, Y. Guo, M. Hayat, and N. Barnes, "Unsupervised primitive discovery for improved 3d generative modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9739–9748.