J Inf Process Syst, Vol.17, No.4, pp.787~800, August 2021
https://doi.org/10.3745/JIPS.02.0160

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Pointwise CNN for 3D Object Classification on Point Cloud

Wei Song*, Zishu Liu*, Yifei Tian**, and Simon Fong**

Abstract

Three-dimensional (3D) object classification tasks using point clouds are widely used in 3D modeling, face recognition, and robotic missions. However, processing raw point clouds directly is problematic for a traditional convolutional network due to the irregular data format of point clouds. This paper proposes a pointwise convolution neural network (CNN) structure that can process point cloud data directly without preprocessing. First, a 2D convolutional layer is introduced to percept coordinate information of each point. Then, multiple 2D convolutional layers and a global max pooling layer are applied to extract global features. Finally, based on the extracted features, fully connected layers predict the class labels of objects. We evaluated the proposed pointwise CNN structure on the ModelNet10 dataset. The proposed structure obtained higher accuracy compared to the existing methods. Experiments using the ModelNet10 dataset also prove that the difference in the point number of point clouds does not significantly influence on the proposed pointwise CNN structure.

Keywords

Point Clouds, Pointwise CNN, 3D Object Classification

1. Introduction

In recent years, using point clouds for three-dimensional (3D) object classification has become a popular method in various fields, such as face recognition [1], 3D modeling [2], intelligence surveillance [3], and robotic missions [4]. The characteristics of point clouds, for example, high precision, are vital for object classification. Compared to traditional two-dimensional (2D) images captured by a camera, point clouds have some significant advantages. For example, point clouds are invariant to lighting, rotation, and color. However, the unordered arrangement, inhomogeneous densities, and non-structural distributions of point clouds present challenges for object classification.

To address these problems, researchers use point cloud features as the basis for classification rather than processing raw point cloud data. Most methods, such as 3D shape context [5], point signature [6], and clustered viewpoint feature histogram [7], focus on analyzing and extracting point cloud features, such as geometric, shape, and structural attributes, or combinations of multiple attributes [8]. Traditionally, data size may increase unnecessarily when input to convolutional networks because convolutional networks require regular inputs. Thus, PointNet [9], a neural network structure that takes

Manuscript received October 11, 2019; first revision January 13, 2020; second revision April 23, 2020; accepted May 29, 2020.

Corresponding Author: Wei Song (sw@ncut.edu.cn)
* School of Information Science and Technology, North China University of Technology, Beijing, China (sw@ncut.edu.cn, 87903631@qq.com)
** Dept. of Computer and Information Science, University of Macau, Macau, China (tianyifei0000@sina.com, ccfong@umac.mo)

the original point clouds as input, has been developed. This paper proposes a lightweight pointwise convolution neural network (CNN) structure for 3D object classification. The proposed pointwise CNN requires fewer training epochs than the PointNet and achieves 87.07% classification accuracy on the ModelNet10 dataset.

The remainder of this paper is organized as follows. Section 2 introduces previous studies to 3D object classification in point clouds. Section 3 describes the proposed network structure. Section 4 describes the experiments and evaluates the classification results. Conclusions and suggestions for future work are provided in Section 5.

2. Related Work

Researchers have done considerable work in object classification using 3D point cloud data. Such studies primarily employed feature-based, graph matching-based, and machine learning-based object classification methods.

In general, feature-based object classification methods can be classified as global and local methods. Methods that combine two categories of features also occur in the face recognition domain [10]. In global feature-based methods, target objects must first be separated from the background. Then, their geometric features are matched to classify the objects. Drost et al. [11] introduced oriented point pair features and employed a fast voting scheme to match models locally. Their proposed method grouped similar features of a model when mapping point pair feature space to the model, which increased the speed of the algorithm when processing point cloud data. Rusu et al. [12] introduced the view feature histogram (VFH), which added viewpoint information to the extended fast point feature histogram (FPFH) information introduced in their previous work [13]. The VFH method could recognize both objects and their pose. In addition, it collected statistics of the relative angle between the surface normal and the direction of the central viewpoint. Therefore, the computational cost of the VFH was reduced, which allowed it to be used in real-time processing. Rusu et al. [14] also proposed global fast point feature histograms to capture the local geometric relationships in an object. In that study, a support vector machine was employed to label and recognize objects, and high classification accuracy was achieved. Marton et al. [15] introduced a global radius-based surface descriptor (RSD) that rasterized the point cloud and sorted the cambers first. Then, the RSD was used to calculate features for classification. Wohlkinger and Vincze [16] introduced an ensemble of shape functions (ESF) for real-time 3D object recognition. This descriptor combined angle, distance between two points, and shape functions, and significantly improved recognition rate. However, experimental results indicated that the ESF descriptor had difficulty identifying similarly shaped objects. Chen et al. [17] proposed the global Fourier histogram (GFH) descriptor, which could use the cylindrical angular coordinate and achieve rotational independence around the vertical axis. Their experimental results indicated that the GFH classified almost all objects correctly; however, it encountered the same problem as ESF, i.e., the descriptor might return incorrect results if two objects had a similar shape. In general, global feature-based methods failed to describe some details of the object and were likely influenced by noise and occlusion.

To address the limitations of global features, researchers focused on developing methods that used local features for classification, for example an object's key points. Spin image (SI) is a classic method introduced by Johnson and Hebert [18]. In SI, 2D spin images were used to represent 3D point clouds.

Although SI showed robustness against noise and occlusion, it required uniform distribution of point cloud data. Guo et al. [19] introduced the Tri-Spin-Image (TriSI) descriptor, which recognized 3D objects under clutter interference and occlusion conditions. Their system included a local reference frame and generated signatures. Then, these signatures were used to compress TriSI features, and a hierarchical feature matching method was employed for object recognition. TriSI showed robustness against noise and varying resolutions, which resolved the weakness of the classic SI method. Rusu et al. [20] introduced a point feature histogram (PFH) and a FPFH [13]. The PFH calculated features, such as the distance between two points and the angle between two points' normal vectors, and then mapped those features to a histogram to obtain statistics. However, the PFH algorithm was highly complex. Thus, the FPFH was developed to improve the efficiency of PFH. FPFH used the relationship between a normal vector and its *k*-neighbor, which made FPFH more efficient than PFH. Salti et al. [21] introduced the signature of histograms of orientation (SHOT) method, which was robust against noise and rotational invariance. Prakhya et al. [22] improved the SHOT method [21] by introducing a "binary" 3D feature descriptor called binary signature of histograms of orientations (B-SHOT). They employed binary quantization to convert a real vector to a binary vector, which made B-SHOT run faster and require less memory than SHOT. He and Mei [23] proposed a new spin image-based registration (SIR) algorithm using 3D feature space that included the Tsallis entropy of the spin image and the laser sensor's reflection intensity. The new SIR algorithm improved the robustness and computational speed of previous SIR algorithms.

With the successful application of deep learning systems in many fields, researchers also focused on applying deep learning systems to 3D data. Maimaitimin et al. [24] converted point clouds to a surface-condition-feature map for feature extraction using an autoencoder. The geometric features extracted work well even with extremely noisy data. Su et al. [25] proposed a multi-view CNN that rendered point clouds into 2D images (views), and then combined information from different views for classification. A view-pooling layer was employed to pool the extracted view-based features from different views. The classification accuracy of their proposed method was significantly better than other 3D shape descriptors. Bobkov et al. [26] developed a point pair descriptor and a four-dimensional (4D) CNN for object classification. The proposed method showed robustness to noise and occlusion in point clouds; however, tuning network hyperparameters required optimization and the computational cost of a 4D convolution cloud was quite high. Ben-Shabat et al. [27] introduced a new representation of a 3D point cloud called 3D modified Fisher vectors. Employing continuous generalized Fisher vectors and a coarse discrete grid structure provided robustness against data loss. Li et al. [28] introduced a new deep neural network structure called a field probing neural network that employed field probing filters to extract features from volumetric space effectively; therefore, it ran much faster than traditional 3D CNNs. Zhou and Tuzel [29] proposed VoxelNet, a 3D detection network that puts a point cloud into an equidistant 3D voxel and then calculates a unified feature for each voxel using a voxel feature encoding layer. Engelcke et al. [30] proposed an efficient CNN to recognize 3D objects in point clouds. They introduced feature-centric voting scheme convolutional layers for recognition. Their model used fewer layers than previous models and simultaneously realized competitive processing time. Fang et al. [31] defined a deep shape descriptor (DeepSD) for a deep neural network. DeepSD demonstrated robustness against noise, incompleteness, and structural variations. Klokov and Lempitsky [32] proposed a deep learning network structure named Kd-network. Differing from traditional convolutional networks, Kd-networks did not require rasterization prior to processing point clouds, which eliminate the time required for rasterization and made the network run faster.

Researchers have also attempted to find ways to handle unordered input data because significant time is required to convert unordered data to ordered data. Vinyals et al. [33] proposed a framework with attention mechanisms that include read, process, and write blocks. Although this framework could handle unordered input sets, their method addressed language processing rather than 3D object classification, and employing that method for 3D object classification tasks was difficult. In this paper, we employed a CNN to classify 3D objects under point cloud data.

3. Pointwise CNN-Based 3D Object Classification Method

We propose a pointwise CNN for 3D object classification (Fig. 1). In our work, raw point cloud data is input to the network directly without any preprocessing. The developed pointwise CNN consists of four convolution layers (Conv1–Conv4), one max pooling layer, and four fully connected layers (FC1–FC4). The Conv1 layer applies 64 1×3 convolution kernels. The other three layers apply 128, 256, and 512 1×1 kernels, respectively. The max pooling layer employs a global max pooling method to extract 512 features from the convolution results of the Conv4 layer. Four fully connected layers, which contain 512, 256, 128, and k neurons in each layer, provide the classification results. A backward process is implemented for training to update the weight matrixes in the convolutional and fully connected layers, and the bias vectors in the fully connected layers. A dropout layer with a drop rate of 0.2 is implemented before FC4 to prevent overfitting. Variable N in Fig. 1 is the number of points.

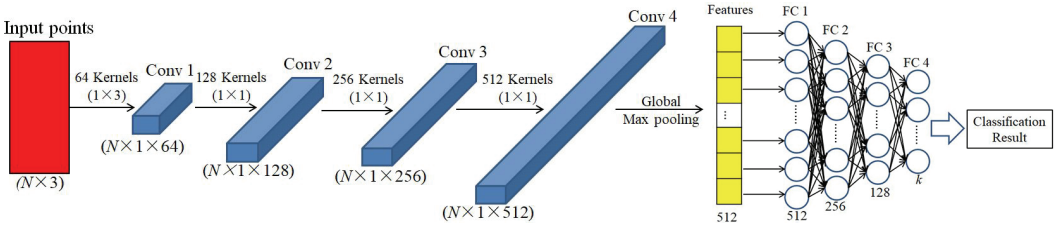

Fig. 1. Proposed pointwise CNN for 3D object classification.

3.1 Forward Process for Training and Testing

Two types of convolutional kernels are used in this study. In the first convolutional layer, we use 1×3 kernels to combine the coordinate information of each point in the point cloud. Fig. 2 and Eq. (1), respectively, illustrate and represents how these kernels work.

In Eq. (1), operator * represents convolution, variable p_n is the coordinate information of a point where n ranges from 1 to N where N is the number of points, $r_{u,n}$ is the result of a convolution, k_u is the u^{th} convolutional kernel where variable u ranges from 1 to m, and variable m is the number of convolution kernels.

$$r_{u,n} = p_n * k_u \tag{1}$$

The size of the output array is calculated in two steps. First, the width and height are calculated using equations $O_w = I_w + 1 - k_w$ and $O_h = I_h + 1 - k_h$, respectively. Second, O_w and O_h are multiplied to obtain the array size. Variables O_w, O_h, I_w, I_h, k_w, and k_h represent the width and height of the output array, input array, and kernel, respectively.

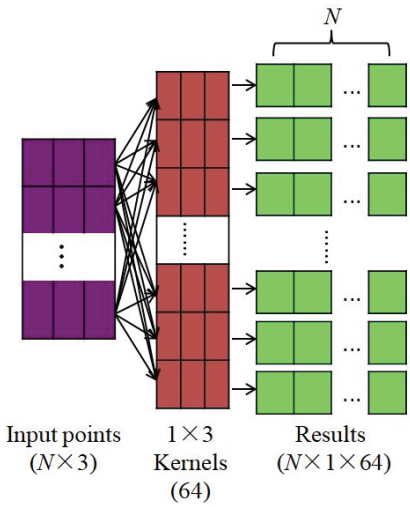

Fig. 2. Convolution computed by 1×3 kernels.

In the other three convolutional layers, 1×1 convolution kernels are used to combine the information from different channels. Each kernel convolutes the values at the same position from different channels in the input arrays. The convolutional results will be saved at the same position in the output array. The output arrays and the kernels are in a one-to-one correspondence. Fig. 3 illustrates how a 1×1 convolutional kernel works where variable *d* represents input and kernel channel.

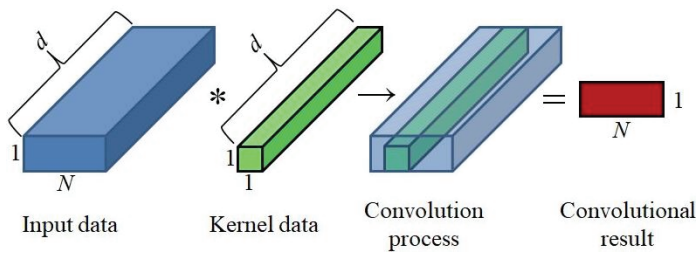

Fig. 3. Convolution computed by the 1×1 kernel.

Note that no biases are added to the output results in all convolutional layers prior to the results being sent to the activation function. The activation function for all four convolutional layers is the leaky ReLU defined in Eq. (2).

$$\sigma(m)=\begin{cases}0.01m, & m\le 0\\ m, & m>0\end{cases} \tag{2}$$

Here, *m* represents the values in output arrays.

To solve the problem of irregular input, we need to utilize a symmetry function as a pooling method. Thus, we employ the global max pooling function to extract 512 features from the last convolutional layer, as shown in Fig. 4. The pool size of this max pooling function equals the size of the output array in the last convolutional layer.

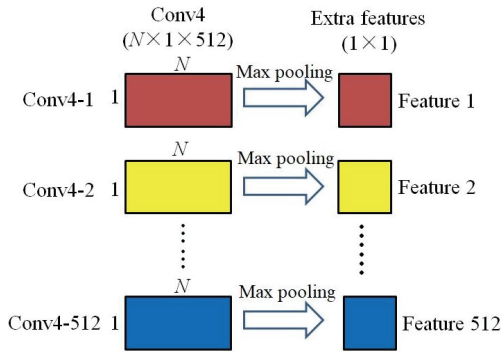

Fig. 4. Global max pooling operation.

The features extracted from the last convolutional layer are then sent to the four fully connected layers. The operational principle of the fully connected layers is shown in Eqs. (3) and (4) where matrix W is a matrix that contains the weights of the layer, function $\sigma(\cdot)$ is the leaky ReLU function defined in Eq. (2), and variables b_i and x_i represent the bias and input vectors in the i^{th} layer, respectively. The activation function for first three fully connected layers is the same as the activation function defined in Eq. (2).

$$z_i = Wx_i + b_i \tag{3}$$

$$a_i = \sigma(z_i) \tag{4}$$

The results of the last layer are then sent to the softmax activation function shown in Eq. (5) where s_i is the result of a softmax operation on a neuron, a_i is calculated by Eq. (4), and variable c is the number of object categories.

$$s_i = \frac{e^{a_i}}{\sum_{j=1}^{c} e^{a_j}} \tag{5}$$

We select the max value from s_i, and then take the number i as the classification result. Note that testing is performed at this stage. However, training requires an updating process for weights and kernels in the network.

3.2 Backward Process

The first step of the backward process used for training is calculating the loss rate q using a loss function. We employ the cross-entropy function shown in Eq. (6), where y_i is an one-hot encoded label for the i^{th} neuron, s_i is the output of this neuron after the softmax operation, and c is the number of categories.

$$q = -\sum_{i=1}^{c} y_i \ln s_i \tag{6}$$

The gradient descent method is implemented to update the weight or the kernel in the network. Thus, we need to calculate the gradient for each layer and then update the weight or the kernel of each layer

accordingly. The gradients of each layer are calculated by the chain derivation rule, which means that the gradient of the last fully connected layer is the most important for updating. Eq. (7) shows how to calculate the gradient of the i^{th} output result z_i in the last fully connected layer. In Eq. (7), variables q and s_j have the same meaning as in Eq. (6), z_i is calculated by Eq. (3), and variable c is the number of categories.

$$\frac{\partial q}{\partial z_i} = \sum_{j=1}^{c} \frac{\partial q}{\partial s_j} \frac{\partial s_j}{\partial z_i} \tag{7}$$

Eqs. (8) and (9) provide the results of two partial differentials in Eq. (7).

$$\frac{\partial s_j}{\partial z_i} = \begin{cases} s_i - s_i^2, i = j \\ -s_i s_j, i \neq j \end{cases} \tag{8}$$

$$\frac{\partial q}{\partial s_j} = -\sum_{j=1}^{c} \frac{y_j}{s_j} \tag{9}$$

Combining the above results, Eq. (7) is equivalent to Eq. (10), where variable c has the same meaning as in Eq. (7) and variable y_i has the same meaning as in Eq. (6). The one-hot encoder is employed in this study; therefore, the gradient of the last fully connected layer can be calculated using Eq. (11) where y_i and s_i have the same meaning as in Eq. (6), and z_i is calculated by Eq. (3).

$$\frac{\partial q}{\partial z_i} = s_i \sum_{j=1}^{c} y_j - y_i \tag{10}$$

$$\frac{\partial q}{\partial z_i} = s_i - y_i \tag{11}$$

The gradients are then used to update the weights and biases in the fully connected layers. The error rates of the last layer and other fully connected layers can be calculated by Eq. (12). Eqs. (13) and (14) show how to calculate the new weight of a layer.

$$e_i = W_{i+1}^T e_{i+1} \circ \sigma'(z_i) \tag{12}$$

$$W_i = W_i - e_i \cdot a_{i-1}^T l \tag{13}$$

$$b_i = b_i - l e_i \tag{14}$$

In Eqs. (12)–(14), operator $\circ$ denotes the Hadamard product, variable e_i is the error rate of the i^{th} layer, W_i is the weight matrix of the i^{th} layer, and variable l is the learning rate for fully connected layers. Variable b_i is the bias vector of the i^{th} layer, a_{i-1} is the input of the i^{th} layer, and z_i is the output of the i^{th} layer before the activation function operation. According to the leaky ReLU function defined Eq. (2), the values of function $\sigma'(z_i)$ are shown in Eq. (15).

$$\sigma'(z_i) = \begin{cases} 0.01, & z_i \leq 0 \\ 1, & z_i > 0 \end{cases} \tag{15}$$

The upsampling operation is done by the max pooling layer, which puts the error rates transmitted from

the previous fully connected layer into the position where the maximum was in the forward process. The blank positions in the pool will be filled with zeroes.

There are two ways to obtain the error rate for convolutional layers in the backward process.

1. If the previous layer is the max pooling layer, the error rate of the previous layer is transmitted to current layer directly.
2. If the previous layer is a convolutional layer, the error rate of the current layer should be calculated based on the error rate of the previous layer. To calculate the error rate, convolution is implemented as shown in Eq. (16). In Eq. (16) e_i is the error rate of the i^{th} layer, k_i are the kernels in the i^{th} layer, operator * means convolution, and function $rot(\cdot)$ means to rotate the kernel 180°. Note, function $\sigma'(z_i)$ and operator ° are defined as in Eq. (12).

$$e_i = e_{i+1} * rot(k_{i+1}) \circ \sigma'(z_i) \tag{16}$$

Eq. (17) shows how the kernels are updated. Here variable k_i is the kernel in the i^{th} convolutional layer, e_i, a_{i-1}, and l are the error rate, input data, and the learning rate for this layer, respectively. Operator * denotes convolution.

$$k_i = k_i - e_i * a_{i-1} l \tag{17}$$

4. Object Recognition Experiments

The proposed network was tested on the ModelNet10 dataset [34]. The dataset contains 4,899 objects from 10 categories, i.e., bathtub, bed, chair, desk, dresser, monitor, nightstand, sofa, table, and toilet, as shown in Fig. 5. Note that the objects in this dataset are aligned in orientation.

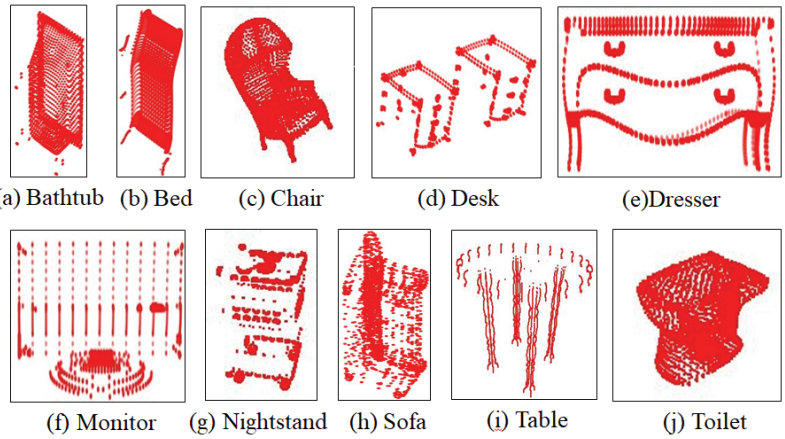

Fig. 5. Examples of categories in the ModelNet10 dataset.

In our experiments, we employed balanced and unbalanced datasets. The partition in the unbalanced set was the same as that of the original ModelNet10 dataset. For the balanced set, we selected 106 samples from the original training set of 10 categories. The remainder of the samples in the original training set were added to the test set. Table 1 shows the partition of training and test sets for each category in the

balanced and unbalanced datasets. The learning rate for all fully connected layers and all convolutional layers was set to 0.0001. The experiments were performed on a computer with an Intel Xeon CPU E5-2670 v3 @ 2.30 GHz, 56 GB RAM, using the Windows 10 operating system.

Table 1. Partition of training sets and test sets

Category	Balanced set			Unbalanced set		
	Training set	Test set	Total	Training set	Test set	Total
Bathtub	106	50	156	106	50	156
Bed	106	509	615	515	100	615
Chair	106	883	989	889	100	989
Desk	106	180	286	200	86	286
Dresser	106	180	286	200	86	286
Monitor	106	459	565	465	100	565
Nightstand	106	180	286	200	86	286
Sofa	106	674	780	680	100	780
Table	106	386	492	392	100	492
Toilet	106	338	444	344	100	444
Total sample	1,060	3,839	4,899	3,991	908	4,899

Table 2. Classification accuracy of each category on test datasets

Category	Accuracy (%)	
	Balanced dataset	Unbalanced dataset
Bathtub	74.00	80.00
Bed	87.22	90.00
Chair	93.54	99.00
Desk	78.33	60.46
Dresser	86.11	87.20
Monitor	96.29	98.00
Nightstand	81.66	86.04
Sofa	88.27	95.00
Table	73.31	79.00
Toilet	94.67	96.00
Average	85.34	87.07

Predict Label

True Label	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night_stand	Sofa	Table	Toilet
Bathtub	37	3	0	1	2	0	0	0	7	0
Bed	12	444	3	15	4	0	1	7	17	6
Chair	0	5	826	0	15	0	14	0	2	21
Desk	5	0	0	141	4	0	8	9	13	0
Dresser	1	0	0	5	155	2	16	0	1	0
Monitor	0	0	0	2	7	442	3	1	4	0
Night_stand	0	0	0	4	26	0	147	0	3	0
Sofa	12	3	0	41	5	0	1	595	17	0
Table	18	2	0	40	6	0	35	2	283	0
Toilet	4	1	6	3	0	0	3	0	1	320

Fig. 6. Confusion matrix of classification results on balanced set.

True Label \ Predict Label	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night_stand	Sofa	Table	Toilet
Bathtub	40	2	0	1	1	0	0	1	5	0
Bed	4	90	1	2	0	0	1	0	2	0
Chair	0	0	99	0	0	0	0	0	0	1
Desk	5	1	0	52	4	1	8	8	7	0
Dresser	0	0	0	2	75	2	6	0	1	0
Monitor	0	0	0	0	2	98	0	0	0	0
Night_stand	0	0	0	0	9	0	74	0	3	0
Sofa	0	1	0	2	1	0	0	95	1	0
Table	3	2	0	8	1	0	7	0	79	0
Toilet	0	0	3	0	0	0	1	0	0	96

Fig. 7. Confusion matrix of classification results on unbalanced set.

We extracted x, y, z coordinate information from the original ModelNet10 Object File Format files and saved the information in individual files that included the coordinate information of a 3D object. In both the training and testing processes, we input the files one at a time. Table 2 shows the classification accuracy of each category on the balanced and unbalanced datasets. With the proposed network, the average accuracy reached 87.07% on the unbalanced dataset after 90 iterations on the training dataset and 85.34% on the balanced set after 140 iterations on the training dataset. The proposed network had a better performance on the unbalance dataset because more samples were used for training.

Figs. 6 and 7 show the classification result confusion matrixes for the balanced and unbalanced datasets, respectively. Note that the values in the matrixes are the number of examples, not percentages. As shown in Fig. 6, the network confused samples in the table and desk classes on the balanced dataset. We thought this occurred because the objects had a similar shape. However, there were some exceptions, e.g., the object in Fig. 5(i) is a round table. As mentioned in Section 2, global features have difficulty classifying objects with similar shape. However, on the unbalanced set, the network misclassified most samples in the desk category as sofa and nightstand. We considered that this result occurred because there are a greater number of samples in the sofa category in the unbalanced training dataset. Another reason for the misclassification was information loss in the point cloud data. The information loss made objects in the desk category share more similarities with samples in the sofa and nightstand categories. This also explained why sofa and nightstand were misclassified as desk (second and third place) on the balanced set.

We compared our results on the unbalanced set against PointNet structure [9] and 3D ShapeNets developed by Wu et al. [34], as shown in Table 3. Compared with these previous studies, the classification accuracy in our work increased by 9.47% and 3.53%, respectively. Compared to the PointNet structure [9], the iteration times on the training set also decreased from 200 to 90.

Table 3. Classification accuracy on test set

Method	Average accuracy (%)	Training iteration
PointNet [9]	77.60	200
3D ShapeNet [34]	83.54	-
Proposed	87.07	90

Fig. 8 is a line chart of the average accuracies for the test set under different training iterations. Here, the horizontal axis is iteration times, and the vertical axis is average accuracy. We found that our proposed

network outperformed the PointNet structure [9] after 20 iterations (81.91%). The average accuracy of the proposed method is almost the same as that of the method developed by Wu et al. [34] after 30 iterations.

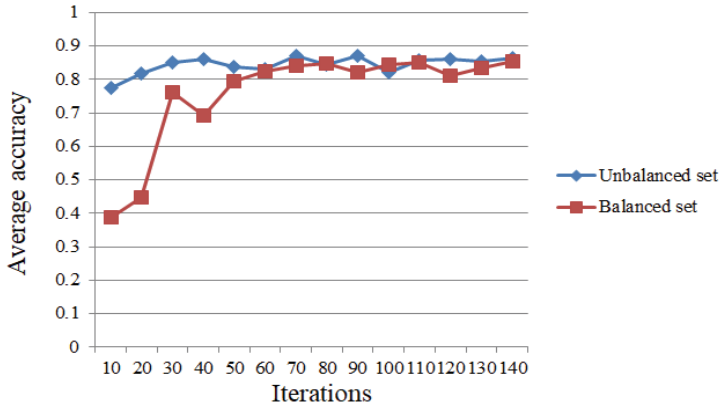

Fig. 8. Testing accuracy under different iterations.

5. Conclusion

This paper proposed a pointwise CNN for 3D point cloud classification that consumes raw point cloud data directly. The proposed system employs four convolutional layers and a max pooling layer to extract features from origin point cloud data. Then, four fully connected layers are employed for classification. The system was tested on the ModelNet10 dataset and achieved average accuracy of 87.07% on the unbalanced training set after 90 iterations and 85.34% on the balanced training set after 140 iterations. Average accuracy improved 9.47% compared to the existing PointNet structure with fewer iterations. However, the proposed network confused objects with similar shape, particularly objects in the desk, nightstand, and table categories. We believe this occurred because we employed global features for object classification and due to information loss in the point cloud data. In future, to improve accuracy, we will consider implementing local features for classification.

Acknowledgement

This research was funded by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program (No. 2020-0-01576) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), National Nature Science Foundation of China (No. 61503005), the Great Wall Scholar Program (No. CIT&TCD20190305), and NCUT funding (No. 110052972027/008).

References

[1] K. W. Bowyer, K. Chang, and P. Flynn, “A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition,” *Computer Vision and Image Understanding*, vol. 101, no. 1, pp. 1-15, 2006.

[2] A. S. Mian, M. Bennamoun, and R. A. Owens, "Automatic correspondence for 3D modeling: an extensive review," *International Journal of Shape Modeling*, vol. 11, no. 2, pp. 253-291, 2005.

[3] M. J. Gomez, F. Garcia, D. Martín, A. de la Escalera, and J. M. Armingol, "Intelligent surveillance of indoor environments based on computer vision and 3D point cloud fusion," *Expert Systems with Applications*, vol. 42, no. 21, pp. 8156-8171, 2015.

[4] Y. Choe, S. Ahn, and M. J. Chung, "Online urban object recognition in point clouds using consecutive point information for urban robotic missions," *Robotics and Autonomous Systems*, vol. 62, no. 8, pp. 1130-1152, 2014.

[5] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Computer Vision – ECCV 2004.* Heidelberg, Germany: Springer, 2004, pp. 224-237

[6] C. S. Chua and R. Jarvis, "Point signatures: a new representation for 3d object recognition," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 63-85, 1997.

[7] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *Proceedings of 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Barcelona, Spain, 2011, pp. 585-592.

[8] W. Hao, Y. H. Wang, X. J. Ning, W. Liang, and Z. H. Shi, "Survey of 3D object recognition for point clouds," *Computer Science*, vol. 44, no. 9, pp. 11-16, 2017.

[9] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "PointNet: a 3D convolutional neural network for real-time object class recognition," in *Proceedings of 2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, Canada, 2016, pp. 1578-1584.

[10] S. Zhou and S. Xiao, "3D face recognition: a survey," *Human-centric Computing and Information Sciences*, vol. 8, article no. 35, 2018. https://doi.org/10.1186/s13673-018-0157-2

[11] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: efficient and robust 3D object recognition," in *Proceedings of 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 2010, pp. 998-1005.

[12] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 2155-2162.

[13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 3212-3217.

[14] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *Proceedings of 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, Kyoto, Japan, 2009, pp. 47-54.

[15] Z. C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D–3D categorization and classification for multimodal perception systems," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1378-1402, 2011.

[16] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *Proceedings of 2011 IEEE International Conference on Robotics and Biomimetics*, Karon Beach, Thailand, 2011, pp. 2987-2992.

[17] T. Chen, B. Dai, D. Liu, and J. Song, "Performance of global descriptors for velodyne-based urban object recognition," in *Proceedings of 2014 IEEE Intelligent Vehicles Symposium Proceedings*, Dearborn, MI, 2014, pp. 667-673.

[18] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433-449, 1999.

[19] Y. Guo, F. A. Sohel, M. Bennamoun, M. Lu, and J. Wan, "TriSI: a distinctive local surface descriptor for 3D modeling and object recognition," in *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications (GRAPP & IVAPP)*, Barcelona, Spain, 2013, pp. 86-93.

[20] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proceedings of 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 2008, pp. 3384-3391.

[21] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251-264, 2014.

[22] S. M. Prakhya, B. Liu, and W. Lin, "B-SHOT: a binary feature descriptor for fast and efficient keypoint matching on 3D point clouds," in *Proceedings of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015, pp. 1929-1934.

[23] Y. He and Y. Mei, "An efficient registration algorithm based on spin image for LiDAR 3D point cloud models," *Neurocomputing*, vol. 151, pp. 354-363, 2015.

[24] M. Maimaitimin, K. Watanabe, and S. Maeyama, "Surface-common-feature descriptor of point cloud data for deep learning," in *Proceedings of 2016 IEEE International Conference on Mechatronics and Automation*, Harbin, China, 2016, pp. 525-529.

[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 945-953.

[26] D. Bobkov, S. Chen, R. Jian, M. Z. Iqbal, and E. Steinbach, "Noise-resistant deep learning for object classification in three-dimensional point clouds using a point pair descriptor," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 865-872, 2018.

[27] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3DmFV: three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145-3152, 2018.

[28] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: field probing neural networks for 3D data," *Advances in Neural Information Processing Systems*, vol. 29, 307-315, 2016.

[29] 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*, Salt Lake City, UT, 2018, pp. 4490-4499.

[30] 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 *Proceedings of 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017, pp. 1355-1361.

[31] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3D deep shape descriptor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 2015, pp. 2319-2328.

[32] R. Klokov and V. Lempitsky, "Escape from cells: deep Kd-networks for the recognition of 3D point cloud models," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 863-872.

[33] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," 2015 [Online]. Available: https://arxiv.org/abs/1511.06391.

[34] 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*, Boston, MA, 2015, pp. 1912-1920.

Wei Song **https://orcid.org/0000-0002-5909-9661**

He received his B.Eng. degree in Software Engineering from Northeastern University, Shenyang, China, in 2005, and his M.Eng. and Dr.Eng. segrees in the Department of Multimedia from Dongguk University, Seoul, Korea, in 2008 and 2013, respectively. Since September 2013, he has been an associate professor at the Department of Digital Media Technology of North China University of Technology. His current research interests are focused on parallel computation, 3D modeling, object recognition, virtual reality, and Internet of Things.

Zishu Liu **https://orcid.org/0000-0002-3444-5919**

He is a postgraduate student at the School of Information of North China University of Technology, Beijing, China, since September 2018. His current research interests are focus on machine learning, 3D object recognition and deep learning.

Yifei Tian **https://orcid.org/0000-0001-8884-2170**

She is a postgraduate student at the Computer and Information Science Department of the University of Macau, Macau, China, since September 2018. Her current research interests are focused on IoT, big data mining, computer graphics, and 3D reconstruction.

Simon Fong **https://orcid.org/0000-0002-1848-7246**

He graduated from La Trobe University, Australia, with a 1st Class Honours B.Eng. Computer Systems degree and a Ph.D. Computer Science degree in 1993 and 1998 respectively. Simon is now working as an Associate Professor at the Computer and Information Science Department of the University of Macau. He is also one of the founding members of the Data Analytics and Collaborative Computing Research Group in the Faculty of Science and Technology. Prior to his academic career, Simon took up various managerial and technical posts, such as systems engineer, IT consultant and e-bommerce director in Australia and Asia.