3D Point Cloud Classification Based on Convolutional Neural Network

Jianrui Lu[1], Wenjing Kang[1], Ruofei Ma[1], and Zhiliang Qin[2(✉)]

[1] Department of Communication Engineering, Harbin Institute of Technology Weihai, Weihai, China
{kwjqq,maruofei}@hit.edu.cn
[2] Technology R&D Center, Weihai Beiyang Electric Group Co. Ltd, Weihai, China
qinzhiliang@beiyang.com

Abstract. With the development of science and technology, the requirements for 3D point cloud classification are increasing. Methods that can directly process point cloud has the advantages of small calculation amount and high real-time performance. Hence, we proposed a novel convolutional neural network(CNN) method to directly extract features from point cloud for 3D object classification. We firstly train a pre-training model with ModelNet40 dataset. Then, we freeze the first five layers of our CNN model and adjust the learning rate to fine tune our CNN model. Finally, we evaluate our methods by ModelNet40 and the classification accuracy of our model can achieve 87.8% which is better than other traditional approaches. We also design some experiments to research the effect of T-Net proposed by Charles R. Qi et al. on 3D object classification. In the end, we find that T-Net has little effect on classification task and it is not necessary to apply in our CNN.

Keywords: 3D object classification · Convolution neural network · Point cloud processing

1 Introduction

In recent years, convolutional neural networks (CNN) has achieved great progress in computer vision. Areas like recognition, classification, detection and segmentation [1–4] has made lots of achievements, especially in two-dimensional (2D). In the field of 2D images, there are already many mature algorithms to complete various signal processing tasks such as recognition and detection. With the development of autonomous driving [5–7] and augmented reality (AR) [8–10], more and more attention were paid on three-dimensional (3D) based convolution and the demand of three-dimensional (3D) point cloud understanding is

This work was supported partially by National Natural Science Foundation of China (Grant No. 61801144, 61971156), Shandong Provincial Natural Science Foundation, China (Grant No. ZR2019QF003, ZR2019MF035), and the Fundamental Research Funds for the Central Universities, China (Grant No. HIT.NSRIF.2019081).

S. Shi et al. (Eds.): 6GN 2021, LNICST 439, pp. 333–344, 2022.
https://doi.org/10.1007/978-3-031-04245-4_29

increasing, especially 3D point cloud classification. Hence, we proposed a novel CNN model to complete 3D point cloud classification task. In general, there are three approaches to process 3D point cloud data. The first method is based on 3D convolution which is a main way to convert 3D point cloud to a regular grid and then directly apply 3D convolution to complete various tasks [13–15]. The second method is based on projection. The first step is to project 3D point cloud to 2D views such as bird-eye-view [16,17]. The second step is to extract features from those 2D views. The third method is based on point which directly process the 3D point cloud. The first and second methods mentioned above usually can achieve good performance, but the amount of calculation is too large and the training time is too long, which make those two method is not suitable for some tasks need high real-time. Hence, in order to have a best performance and high real-time, our proposed CNN model is based on the third method.

For point-based method, the most classic method is PointNet [13] proposed by Charles R. Qi, Hao Su, et al. PointNet can directly consumes point clouds so that it is highly efficient. Besides, the authors of PointNet proposed a mini-network which called T-Net to achieve transformation invariance and batch normality, which is helpful to improve the performance of 3D point cloud segmentation. With reference to the network structure of PointNet, we proposed a novel CNN model. There are totally seven convolution blocks in our model and each block contains one convolution layer and a Batch Normalization layer. ReLU function is selected to be a activation function after every convolution block. The complete network structure will be introduced in detail in the third section.

In our experiment, we firstly train our model (called pre-training model) with ModelNet40 dataset and then adjust the learning rate to fine tune our pre-training model. Fine tuning is a common approach to improve the performance of classification system. According to different size of the target dataset and the similarity between the source dataset and the target dataset, there are different methods to fine tune the model to better fit our training data. How to fine tune a model in different cases will be introduced in detail in Sect. 3. For our model, we freeze the first five layers of our extractor and adjust the learning rate to better train our model. In the end, the classification accuracy of our model can achieve 87.8%, which is better than traditional 3D point cloud classification methods.

In addition, to research the effect of T-Net in 3D point cloud classification, we add two T-Net in our proposed CNN. One is after the first convolution block to adjusts the view of 3D object and achieve transformation invariance. The other one is after the third convolution block to batch normalize data for speeding up the training processing. However, we find that T-Net has little effect on classification task but increase the training time. Hence, in the final version, we do not apply T-Net in our CNN model.

All in all, the contributions of our paper are as follow: 1) we proposed a novel CNN model for 3D object classification task and achieve a high accuracy which is better than other traditional classification methods. 2) we apply T-Net in our CNN model and find that T-Net has little effect on classification task which mean you can replace T-Net to decrease your training time.

2 Related Works

3D Point Cloud Processing: Point cloud is a collection of points sampled from the surface of a 3D object. 3D point cloud is usually used to represent data whose latitude is lower than the latitude of the background space such as curved surfaces in the space. In general, a 3D point cloud is denoted as $P = \{p_i | i = 1, 2, ..., n\}$, where each point p_i is a vector that contains the (x, y, z) coordinates and possibly some other features, e.g., colors, intensity.

Traditional convolution operations are order-dependent and depended on spatially-local correlation. Hence, CNN is effective for data, such as images, that is order and has spatially-local correlation. In recent years, lots of CNNs have been proposed to complete kinds of 2D object tasks and have achieved good performance, e.g., Xception [11], VGG-19 [12]. However, the data represented in point cloud form is unordered and irregular. It is mean that the traditional convolution operations are not suitable for extracting directly features from point cloud. We need to find other approaches to deal with point cloud. At present, methods for point cloud processing mainly are as follow:

- 3D convolution-based method
- projection-based method
- point-based method

3D convolution-based method is one of the main approach to process point cloud. In this method, point cloud data are firstly rasterize into regular grids (called voxels). This operation is called as voxelization. Then, 3D convolution is applied on those regular grids to complete various tasks, e.g., classification, segmentation [13–15].

Projection-based method is based on 2D convolution to extract features from 2D views projected by 3D point cloud. The bird-eye-view projection [16,17] and the spherical projection [18,19] are often used in point cloud processing. Besides, 3D point cloud can be processed with multi-view representation [20,21]. Hang Su et al. proposed a multi-view convolutional neural network (MVCNN) [20] to combine the same 3D shape images from different perspectives to extract 3D shape description operators and used a new structure called view-pooling layer to complete effectively recognition task.

In point-based method, 3D point cloud is directly processed. The most classic method is PointNet [13] proposed by Charles R. Qi, Hao Su, et al. Unlike the two methods mentioned above, PointNet provides a unified architecture to directly consumes point clouds so that the original data can avoid unnecessarily voluminous and some problems. Besides, due to directly processing 3D point cloud, PointNet is highly efficient. Nowadays, PointNet is widely applied in all kinds of applications ranging from 3D object classification, segmentation, to semantic parsing.

PointNet: According to Charles R. Qi et al., inputs of PointNet are unordered point sets. The jth point cloud is represented as $P_j = \{p_i | i = 1, 2, ..., n\}$, where

each point p_i is a vector that only contains the (x, y, z) coordinates. For the point cloud classification task, points are multiplied by a transformation matrix predicted by a mini-network (T-Net in Fig. 1). Two T-Nets are used in PointNet, the first T-Net is used to achieve transformation invariance and the second T-Net is used to batch normal data. Besides, PointNet uses multi MLP to extract the global feature of point cloud and at the last layer, one MLP is used to output k scores for all the k candidate classes. The detailed network architecture is shown in Fig. 2.

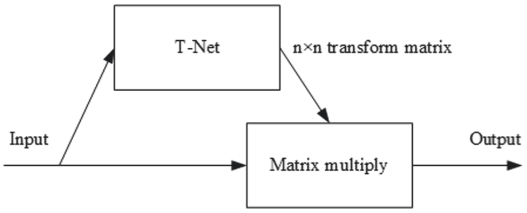

Fig. 1. T-Net

3 Methodology

Our Model: PointNet, first applied input and feature transformations to the input points, has been successfully used in different applications to achieve better performance [13].

With reference to PointNet, we proposed a novel CNN for 3D object classification task. For our model, a point cloud, including lots of points, is given as input. A point cloud is denoted as $P_j = \{p_i | i = 1, 2, \cdots, n\} (j = 1, 2, \cdots, m)$, where p_i means the ith point of the jth point cloud and it is a vector that only contains the (x, y, z) coordinates. m and n mean the number of point clouds and the number of points contained by each point cloud, respectively. Besides, each point cloud data has a label $y_j \in \{1, 2, \cdots, c\}$ where c is the number of classes. We firstly randomly rotated and jittered the data to finish data augmentation. Then, we used a CNN ($ConvNet$) to extract the global feature of inputs and then used a classifier f to predict the label of input point cloud. Formally, we have $y_j = f(ConvNet(P_j))$. At the end, our model will output a $(1 \times c)$ vector $O_j = \{o_1, o_2, \cdots, o_c\}$ which contains the probability that the jth point cloud data belongs to each category. The whole system structure and the neural network training process is shown in Fig. 3.

Our proposed ConvNet mention above has the similar structure as the PointNet [13]. There are seven convolutional blocks and each convolutional block consists of one convolution(Conv) layer with convolution filters and a batch normalization (BN) layer. BN layer is used to speed up our learning processing. After Conv layer and BN layer, The Rectified Linear Units (ReLu) function is used to achieve non-linearity.

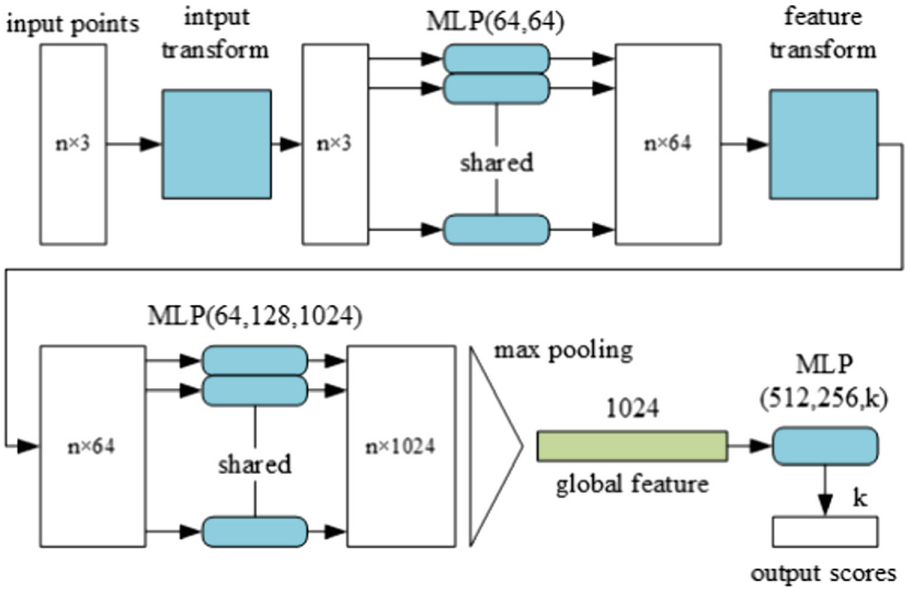

Fig. 2. PointNet

After all Conv layers, max-pooling is performed and the output of max-pooling layer is reshape with size of (,1024). In the end, we used dropout layers (the probability is set to 0.3) to decrease the possibility of over-fitting and fully connected (FC) layers followed by a softmax function to get the score of each 3D object class and classify the point cloud data. The whole architecture of our proposed ConvNet is described in Table 1.

Fine Tuning: In general, when we have a model, we do not simply use this model just as a feature extractor. Adjusting the model to better fit our training data is necessary. For model adjusting, fine tuning is a excellent approach to fit the training data for better performance and this is also the most commonly used. According to different characteristics of our training data, there are different situations of fine tuning. We can denote the data set of source domain as D_S and the data set of target domain as D_T. Our network can be divided into two parts, the first part is a feature extraction denoted as E and the second part is a classifier denoted as C. When our D_T is small but has a high similarity to the D_S used by the pre-trained model, we can only retrain the classifier C. Because of the small size of D_T, there is a risk of over-fitting during retraining E. However, due to the similarity between D_T and D_S is high, both the local feature and global feature are relatively similar. What mean is that the original feature extractor E can do well in D_T and we just need to retrain the C to suit our task. When the D_T is small and it is very different from the D_S, fine tuning is not suitable. In this case, a better solution is generally to extract features from

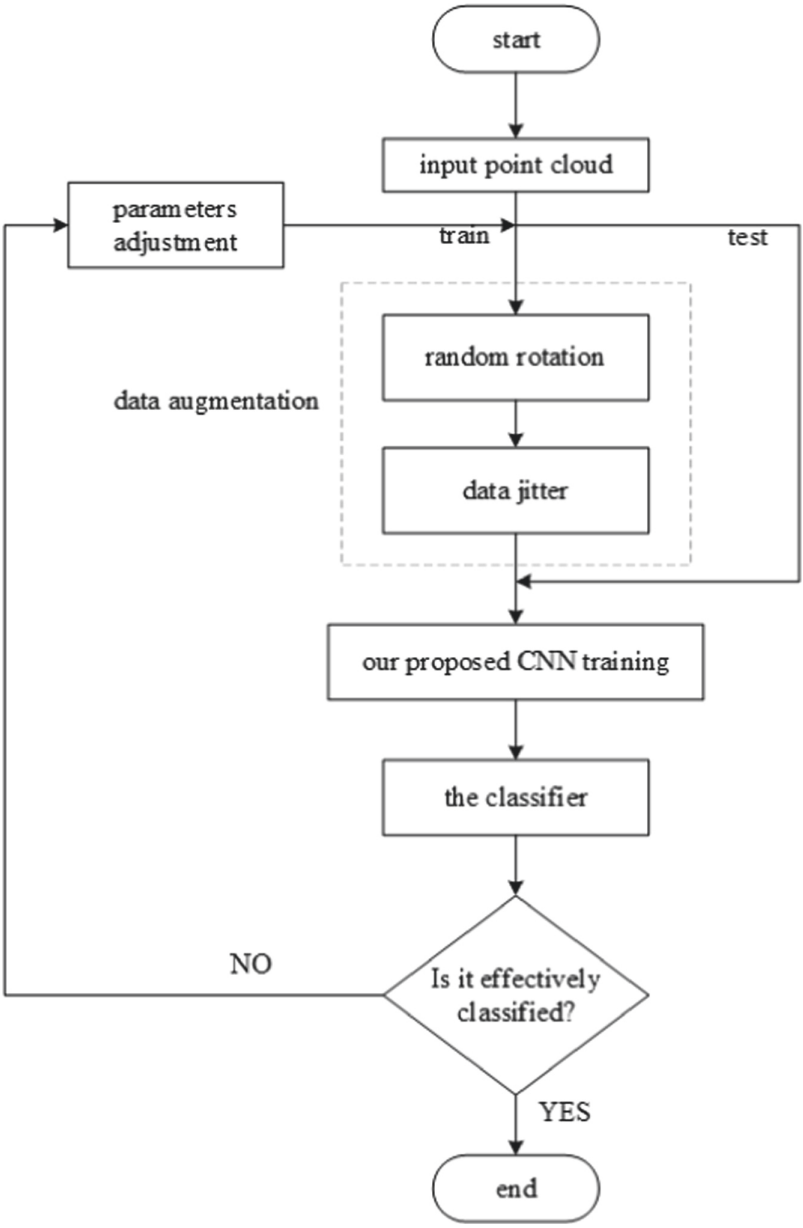

Fig. 3. System structure diagram

a certain layer of E and then train a support vector machines (SVM) classifier. When the D_T is large but it is very different from the D_S, fine tuning is also not suitable. However, due to the large size of the D_T, we can train a model from

Table 1. CNN architecture

Layer	Output shape	Parameter
Input layer	(,2048,3)	0
Conv1-64,BN,ReLU	(,2048,64)	512
Conv1-64,BN,ReLU	(,2048,64)	4416
Conv1-64,BN,ReLU	(,2048,64)	4416
Conv1-128,BN,ReLU	(,2048,128)	8832
Conv1-256,BN,ReLU	(,2048,256)	34048
Conv1-512,BN,ReLU	(,2048,512)	133632
Conv1-1024,BN,ReLU	(,2048,1024)	529408
Max Pooling layer	(,1,1024,1)	0
reshape layer	(,1024)	0
Dropout layer	(,512)	0
FC layer	(,256)	131328
Batch Normalization layer	(,256)	1024
Dropout layer	(,256)	0
FC layer	(,Number of Classes)	10280

scratch or train all layers of E and C. When the D_T is large and has a high similarity to the D_S, we do not have to worry about over-fitting, so it is better to fine tune the entire network structure.

In our experiment, we firstly used ModelNet40 (a dataset contained 12311 CAD models from 40 object categories) [22] to train our CNN model. Then we also used ModelNet40 to fine tune our model. For feature extractor E, we froze the first five layers which only extract some basic local features and updated the later layers of our CNN structure. This is because the features obtained in the first few layers of the network are more basic such as edge and corner feature, which are easily applied to all tasks. Hence it is not necessary to retrain the first few layers. Beisdes, the higher the network layer number, the closer the connection with the data set global information and those global features are closely related to the original data.

4 Experiment and Results

Experiment Settings: ModelNet40 is used in our experiment to evaluate our proposed CNN. There are totally 12311 CAD models from 40 man-made object categories [22]. The dataset is split into two groups: 9843 samples for training and 2468 for testing. For this experiment, Processor used is Intel Core i5-10400F, Graphics card in this system is NVIDIA GeForce GTX 1660 Ti. The software used in this experiment is as follows: python with the version of 3.7.6, keras 2.2.4 and tensorflow 1.14.0 with GPU. For the pre-training process, the learning rate is set to 0.0003 and the batch size is set to 16.

Result: In our experiment, we train and fine tune our CNN model from 10 epoch to 200 epoch, the training accuracy and the validation accuracy is shown in Fig. 4. As shown in Fig. 4, no matter training accuracy and validation accuracy are increasing with number of epoch going up. While the number of epoch is up to 75, the two accuracy curves start to converge. In the end, our model can achieve a best performance on validation set (87.8%) and the number of epoch is 177. Figure 5 shows the training and validation loss. We can know from Fig. 5 that training loss and validation loss were decreased to below 0.2 when the training epoch is up to 60. This performance is shown us our method is effective. As shown in Table 2, we compare our proposed CNN model to some traditional methods based on 3D-convolution or projection. It is seen from Table 2 that our system has a better performance compared with other methods and the accuracy of our proposed CNN model can achieve 87.8%. Our model obtains an

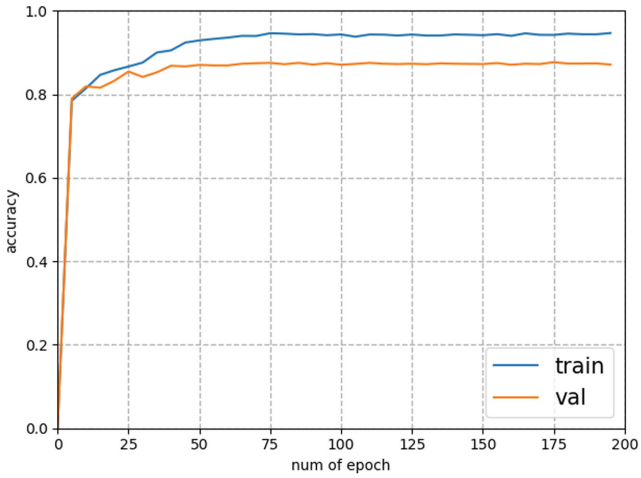

Fig. 4. Training and Validation accuracy

Table 2. The overall compare of performance on ModelNet40

Model	Average accuracy
3D-GAN [23]	83.3%
VoxNet [24]	83%
3DShapeNets [26]	77%
Primitive-GAN [25]	86.4%
Xu and Todorovic proposed [27]	81.26%
ECC [28]	83.2%
Ours	**87.8%**

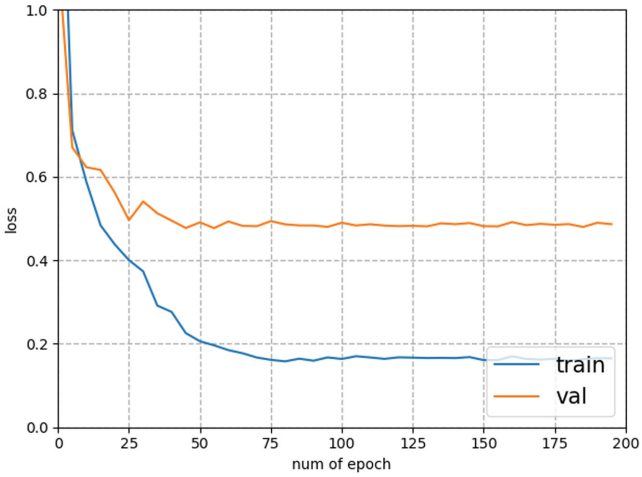

Fig. 5. Training and Validation loss

improvement of 4.5%, 4.8%, 10.8%, 1.4%, 6.54% and 4.6% to other methods, respectively. In addition, we want to research if the T-Net, proposed by Charles R. Qi et al., will improve the performance of our classification system. Hence, we apply two T-Net blocks in our CNN model. The first T-Net block is used after the first Conv block to eliminate the viewing angle of the 3D object and achieve transformation invariance. The second T-Net block is used after the third Conv block to finish batch normalizing. The comparison between system using T-Net and not using T-Net is shown in Fig. 6 and the training time of each epoch is shown in Table 3.

Table 3. The training time of different system

System using T-Net	System not using T-Net
140 s/epoch	70 s/epoch

From Fig. 6, we can see that the accuracy curve of the system using T-Net is similar to the accuracy curve of the system not using T-Net, which mean that T-Net has little effect on our CNN model for classification task. However, it takes 140 s per epoch to train the model if T-Net is used in the system, which is 50% more time than a system that does not use T-Net. Hence, T-Net is not a good choose for our system to complete the 3D object classification task.

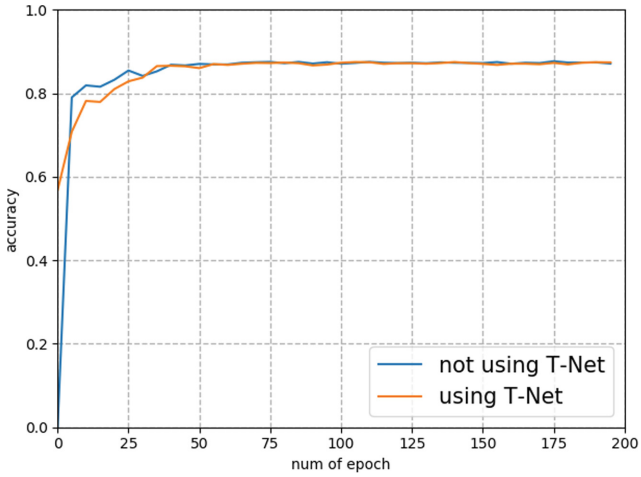

Fig. 6. Validation accuracy of the system not using T-Net and using T-Net

5 Conclusion

In this work, we propose a novel convolutional neural network that directly consumes point cloud. Our network was firstly trained on ModelNet40 and then using the same dataset to fine tune the model. Via some appropriate training, our model achieved a better accuracy (87.8%) than other traditional approaches. As shown in the result of our experiments, the proposed CNN model could be applied in other 3D object tasks such as 3D object segmentation.

References

1. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014). https://doi.org/10.1109/CVPR.2014.81
2. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
3. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR, January 2015
4. Malbog, M.A.: MASK R-CNN for pedestrian crosswalk detection and instance segmentation. In: 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), pp. 1–5 (2019). https://doi.org/10.1109/ICETAS48360.2019.9117217
5. Behley, J., et al.: SemanticKITTI: a dataset for semantic scene understanding of LiDAR sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)

6. Caesar, H., et al.: nuScenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11621–11631 (2020)
7. Yue, X., Wu, B., Seshia, S.A., Keutzer, K., Sangiovanni-Vincentelli, A.L.: A LiDAR point cloud generator: from a virtual world to autonomous driving. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, pp. 458–464 (2018)
8. SketchUp: 3D modeling online free 3D warehouse models (2021). https://3dwarehouse.sketchup.com
9. Wu, Z., et al.: 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015
10. Shi, B., Bai, S., Zhou, Z., Bai, X.: DeepPano: deep panoramic representation for 3-D shape recognition. IEEE Sig. Process. Lett. **22**(12), 2339–2343 (2015)
11. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, pp. 1800–1807 (2017)
12. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv e-prints arXiv:1409.1556, September 2014
13. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
14. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: convolution on x-transformed points. In: Advances in Neural Information Processing Systems, pp. 820–830 (2018)
15. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. arXiv preprint arXiv:1703.06870 (2017)
16. Yang, B., Luo, W., Urtasun, R.: PIXOR: real-time 3D object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7652–7660 (2018)
17. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12697–12705 (2019)
18. Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K.: SqueezeSegV2: improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In: ICRA (2019)
19. Xu, C., et al.: SqueezeSegV3: spatially-adaptive convolution for efficient point-cloud segmentation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12373, pp. 1–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58604-1_1
20. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 945–953 (2015). https://doi.org/10.1109/ICCV.2015.114
21. Liang, Q., Wang, Y., Nie, W., Li, Q.: MVCLN: multi-view convolutional LSTM network for cross-media 3D shape recognition. IEEE Access **8**, 139792–139802 (2020). https://doi.org/10.1109/ACCESS.2020.3012692
22. Wu, Z., et al.: 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920 (2015)

23. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: NIPS (2016)
24. Maturana, D., Scherer, S.: VoxNet: a 3D Convolutional Neural Network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928 (2015). https://doi.org/10.1109/IROS.2015.7353481
25. Khan, S.H., Guo, Y., Hayat, M., Barnes, N.: Unsupervised primitive discovery for improved 3D generative modeling. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9731–9740 (2019). https://doi.org/10.1109/CVPR.2019.00997
26. Zhirong, W., et al.: 3D ShapeNets: a deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1912–1920 (2015). https://doi.org/10.1109/CVPR.2015.7298801
27. Xu, X., Todorovic, S.: Beam Search for Learning a Deep Convolutional Neural Network of 3D Shapes. arXiv e-prints arXiv:1612.04774 (2016)
28. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs (2017). https://arxiv.org/abs/1704.02901