PointConv++: A muti-resolutional network for point cloud classification

Qi Yang,[1] **Yanan Lin,**[1] **Jiawei Li,**[1] **Yifan Wang,**[1] **Fenghong Yang**[1]
[1]Computer Science Department, Xiamen University, China
yangqi@stu.xmu.edu.com, 739559801@qq.com, jiaweili98@163.com, 1246410140@qq.com, xmuyfh@qq.com

Abstract

Point cloud classification is a 3D task for point labeling, which is widely used in autonomous driving, augmented reality, robotics, etc. Since PointConv was proposed, deep convolution on point sets has been the concentration of 3D research. However, existing convolutional network on point cloud usually generate features regardless of density unevenness of the point cloud. In this work, we propose a novel network called PointConv++ for 3D object classification. With hierarchical convolution and communication among multiple resolutions, PointConv++ is able to extract robust features and generalize to point sets with varying density. One key process of PointConv++ is the recursive application of PointConv on the point cloud. With the learning of weight functions and density estimation, convolution operation is simulated for local feature extraction. Another key process of PointConv++ is the exploitation of multi-resolution(MR) block. By broadcasting information across multiple resolutions, MR block allows layers in different resolution branches to exchange information with each other, which benefits the robust extraction of global features as well as the integration of contextual information. Experiments show that PointConv++ is able to learn deep point set features efficiently and robustly. In classification task of point cloud, PointConv++ network performs better than or comparable with the state-of-the-art approaches.

Introduction

A point cloud is a set of data points in space. The points represent a 3D shape or object. Each point has its set of X, Y and Z coordinates. Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. Figure 1 shows a point cloud from a laser scanner.

With the developments of laser radar and other imaging instruments, three-dimensional (3D) data is becoming much more easily available. Consequently, the effective processing and analysis methods should be investigated for 3D-related applications. As the representative 3D data, point cloud has been widely adopted in indoor navigation, autopilot, and augmented reality, etc. The effective classification of point clouds can be helpful for the better understanding of intelligent systems to different complicated environments.

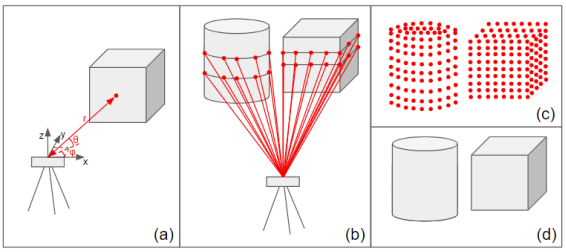

Figure 1: A point cloud from a laser scanner.

Therefore, the accurate classification of point clouds plays an important role in related practical applications.

With the improvement of computing power and the substantial increase of data, deep learning (LeCun, Bengio, and Hinton 2015) has become more and more popular for point cloud classification (Maturana and Scherer 2015; Qi et al. 2016; Su et al. 2015). State-of-the-art point cloud classification methods are mostly based on deep neural networks. Figure 2 shows the basic structure of point cloud classification based on deep learning. Points in a point cloud are irregular and unordered so they cannot be easily handled by regular 2D CNNs. To address this problem, PointNet (Qi et al. 2017a) uses multi-layer perceptrons (MLPs) to extract features for each point separately. Then, it is followed by a symmetric function to accumulate all point features. Subsequent methods, including (Qi et al. 2017b; Wang et al. 2019; Shen et al. 2018), focus on effectively processing the information of neighboring points jointly rather than individually. PointNet++ (Qi et al. 2017b) utilizes the PointNet in sampled local regions and aggregates features hierarchically. DGCNN (Wang et al. 2019) builds dynamic connections among points in their feature level and updates point features based on their neighboring points in the feature space.

Related Work

3D data has multiple popular representations, leading to various approaches for learning. The volumetric representation encodes a 3D shape as a 3D tensor of binary or real values. The multi-view representation encodes a 3D shape as a collection of renderings from multiple viewpoints. The mesh representation encodes a 3D shape as a collection of

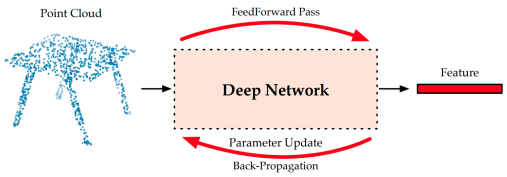

Figure 2: The basic structure of point cloud classification based on deep learning. Point cloud data are fed into deep neural networks in the feedforward pass and errors are propagated in the backward direction. This process is conducted iteratively until convergence. Labels are needed to update all model parameters.

points, normal vectors and faces. Volumetric CNNs: (Wu et al. 2015; Maturana and Scherer 2015; Qi et al. 2016) voxelizes 3D point clouds into volumetric grids. The point cloud data can be regularized and processed using convolutional neural networks. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. (Riegler, Osman Ulusoy, and Geiger 2017) improves the resolution significantly by using a set of unbalanced octrees where each leaf node stores a pooled feature representation. Multiview CNNs: (Su et al. 2015; Qi et al. 2016) proposed to project 3D point clouds or shapes into several 2D images, and then apply 2D convolutional networks for classification. Although this line of methods has achieved dominating performances on shape classification and retrieval tasks (Savva et al. 2016), it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. Spectral CNNs: Some latest works (Bruna et al. 2013; Masci et al. 2015) use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture.

In contrast to work converting irregular 3D point clouds to 2D images or 3D voxels that may cause loss of imformation, some work (Qi et al. 2017a; Ravanbakhsh, Schneider, and Poczos 2016; Hua, Tran, and Yeung 2018) directly use raw point cloud as input. However, (Qi et al. 2017a) directly uses MLP and max pooling to extract features from the whole point cloud, it lacks the ability to capture the local structure. PointNet++ (Qi et al. 2017b) introduces a hierarchical structure, it samples and groups the point clouds into small neighborhoods, then uses pointNet to extract features from each neighborhoods. This hierarchical structure improved the PointNet's ability to extract local structures, it is analogue to the multiple convolution layers in CNN that extract features from local structures and forms an overall feature. However, PointNet++ still uses max pooling to obtain features that only keep the strongest activation on features in a region, which may lose some detailed information. Some researchers tries to extend the 2D convolution operation to 3d point clouds. The irregularity of point cloud makes it difficult to use convolution. (Su et al. 2018) implemented sparse bilateral convolution in point cloud. PointCNN (Li et al. 2018) learns a χ-transformation from the input points, the χ-transformation can weight the input features assosiated with the points and permutate the points into a latent and potentially canonical order. But PointCNN lacks the ability to achieve permutation-invariance, which is important for point clouds. (Jia et al. 2016) proposed a method to treat the weight filter in 2d convolution as a continuous function, which can be approximated using MLP. (Simonovsky and Komodakis 2017) firstly introduced the idea into 3d graph structure. (Lu et al. 2020) PointConv extend the dynamic filter to a new convolution operation to approximate the 3D continuous convolution. It achieves permutation-invariance and translation invariance, and density information is considered to reweight the convolution. (Le, Kokkinos, and Mitra 2020) Extended the multi-resolution grouping of (Lu et al. 2020) by adding Cross Links between different resolutions, which lead to the increase in both training speed and performance. However, (Le, Kokkinos, and Mitra 2020) still uses max pooling to capture features in local regions, which causes the loss of detailed information. Our work implements multi-resolution grouping and cross links in (Le, Kokkinos, and Mitra 2020) and the convolution methond in (Lu et al. 2020), can both capture muti-resolution information and avoids the loss of detailed information caused by maxpooling, is different from all the above methods.

Proposed Solution

Review of Multi-resolution

In PointNet (Qi et al. 2017a), given an unordered point set $\{x_1, x_2, \ldots, x_n\}$ with $x_i \in R^d$, we can define a set of function $f : \chi \rightarrow R$ that maps a set of points to a vector : $f(x_1, x_2, \ldots, x_n) = \gamma(MAX_{i=1}\{h(x_i)\})$ where γ and h are usually multi-layer perception(MLP) networks. And the set function f is invariant to input point permutations. PointNet achieved impressive performance on a few benchmarks, but it lacks the ability to capture local context at different scales. To solve the problem, PointNet++ (Qi et al. 2017b) builds a hierarchical grouping of points and progressively abstract larger and larger local regions along the hierarchy. The hierarchical structure is composed by a number of set abstraction levels, each set abstraction level is made of three key layers: Sampling layer, Grouping layer and PointNet layer. As we know, it is common that a point set comes with non-uniform density in different areas. So PointNet++ introduces the Multi-resolution grouping(MRG) which summarizes the features from different levels. As shown in Figure 3, the two vectors concatenated, one of them is obtained from the lower level and the other one is obtained directly from all point s. The fusion of local, fine-grained information and global, semantic-level context can boost the discriminative power of the resulting features (Le, Kokkinos, and Mitra 2020).

Revisit PointConv

Previous works (Qi et al. 2017a,b) use maxpooling and MLPs to extract features from point clouds, which will cause loss of information. It is intuitive to extend the convolution operation in 2D image tasks to 3D point clouds. However, point clouds are unordered and do not conform to the regular

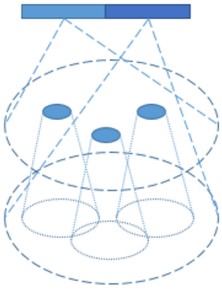

Figure 3: Multi-resolution grouping(MRG)

lattice grids as in 2D images, which makes it hard difficult to implement 2D convolution directly.

The work of (Wu, Qi, and Fuxin 2019) firstly go back to the continuous 3D convolution operation, which can be written as Equation 1.

$$Conv(W,F)_{xyz} = \iiint_{(\delta_x,\delta_y,\delta_z)\in G} W(\delta_x,\delta_y,\delta_z)\, F(x+\delta_x, y+\delta_y, z+\delta_z)\, d\delta_x\delta_y\delta_z \quad (1)$$

where $F(x+\delta_{x'}y+\delta_{y'}z+\delta_z)$ is the feature of a point in the local region centered around point $p=(x,y,z)$. A point cloud canbe viewed as anon-uniform sample from the continuous $\mathbb{R}^3$ space. In each local region, $(\delta_x,\delta_y,\delta_z)$ could be any possible position in the local region. PointConv proposes a special convolution operation which can be written as Equation 2.

$$PointConv(S,W,F)_{xyz} = \sum_{(\delta_x,\delta_y,\delta_z)\in G} S(\delta_x\delta_y,\delta_z)\, W(\delta_x,\delta_y,\delta_z)\, F(x+\delta_x, y+\delta_y, z+\delta_z) \quad (2)$$

Where $S(\delta_x,\delta_y,\delta_z)$ is the inverse density at point $(\delta_x,\delta_y,\delta_z)$. Because of the density of points varies across the whole point cloud, the density information added can reduce the contribution of points in dense areas. The inverse density can be caculated using kernelized density estimation. The weight function $W(\delta_x,\delta_y,\delta_z)$ can be approximated from the 3D coordinates $(\delta_x,\delta_y,\delta_z)$ using multi-layer perceptrons.

The paper (Wu, Qi, and Fuxin 2019) also proposed a novel reformulation to implement PointConv by reducing it to two standard operations: matrix multiplication and 1x1 convolution. This reformulation is less memory consuming and more efficient. Figure 4 shows the efficient version of PointConv.

Introduction of our model

In this section we first introduce our MRG method, then we introduce our MultiReolution (MR) block, and the detail of the implementation of feature combination.

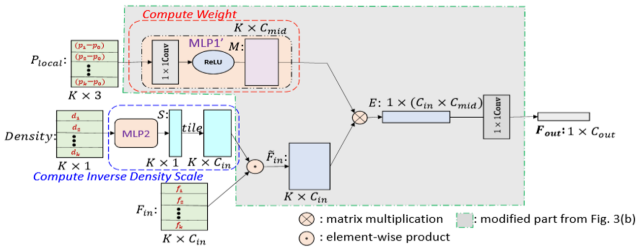

Figure 4: The efficient PointConv structure in one local region proposed in (Wu, Qi, and Fuxin 2019). MLP1' is used to approximate the weight function, MLP2 is used to compute Inverse Density Scale.

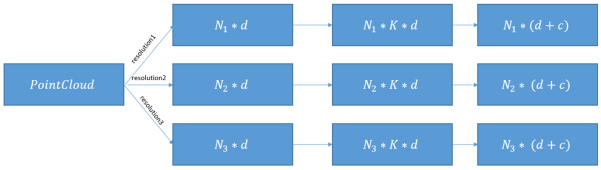

Figure 5: The processing of different resolutions.

Design of MRG We use three different resolutions in our network to obtain features of different scales. Assume that the number of points sampled at the three resolutions are N_1 , N_2 and N_3 respectively. After passing through a MR block, we get three features that respectively carry information of N_1, N_2, N_3 regions. The processing of different resolutions is shown in Figure 5. Input the original point cloud data and sample with different resolutions to extract features of different scales.

MutiReolution PointConv In this section we propose our MutiReolution (MR) block as Figure 6 shows, which takes a point cloud as input then sample and group the input into different resolution. The sampling operation uses Farthest Point Sampling (FPS) used in (Qi et al. 2017b) to pick out centroid points. the grouping operation takes the k nearest points from every centroid points and group them into one local region. The PointConv takes points in every local region and extract their features. After the PointConv operation, only the centroid points are passed to the next MR block, so that a hierarchical structure shown in Figure 7 can be formed.

As it is shown in Figure 7. After every MR block, the points are lesser and feature becomes larger. After the finnal block, there will be only one point left and features from the whole point cloud is combined in one 1x1024 feature. The final feature is passed to a MLP to classify the point cloud.

Feature combination After getting features from different resolutions, we need to merge them for the next step of processing. We have two ways to accomplish this. First, we can use a fully connected layer to transform the feature dimensions according to our needs. Second, we can achieve upsampling from low resolution to high resolution by linear

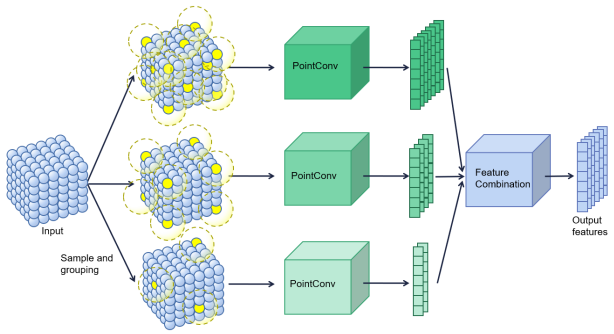

Figure 6: Our MR Block: We first sample and group the point cloud into different resolution, then use PointConv to extract features in different local regions. Then we use a bunch of SLPs to combine the features in different resolution.

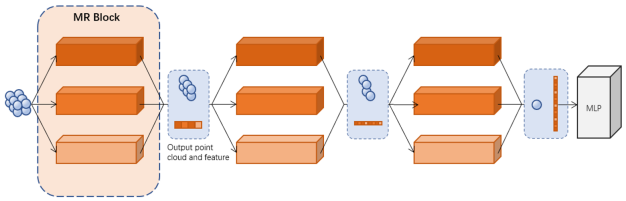

Figure 7: The hierarchical network constructed using our MR blocks.

interpolation in the spatial space using the $K_u = 3$ closest neighbors. After getting back to the original resolution, we can easily combine features by concatenation or addition. These two ways are shown in Figure 8.

Experiments

In order to evaluate our new PointConv++ network, we conduct experiments on the widely used dataset ModelNet40. In all experiments, we use the PyTorch framework to implement the proposed PointConv++ method, and a single GTX TITAN X is used for training and testing. The batch size is set to 32. ReLU and batch normalization are applied after each layer except the last fully connected layer. We use the SGD optimizer to minimize the loss. The Total training takes 50 epochs. The learning rate is set to 0.0005. Firstly, we introduce the dataset we used to conduct the experiments. Secondly, we compare the proposed PointConv++ method with several state-of-the-art point cloud classification methods. Thirdly, we perform ablation studies to evaluate the key components of the proposed PointConv++ method. Finally, we visualize the output of MR block, which conduct multi-resolution sampling on original data.

Dataset

ModelNet40 contains 12,311 CAD models of 40 categories (mostly man-made). We use the official split with 9,843 shapes for training and 2,468 for testing. For fair comparison, we employ the same data augmentation strategy as Pointnet by randomly rotating the point cloud along the z-axis and jittering each point by a Gaussian noise with zero mean and 0.02 standard deviation. Figure 9 show some samples of ModelNet40 dataset and how data augmentation is employed.

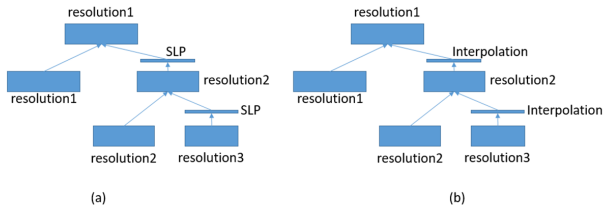

Figure 8: (a)Use SLP to upsample. (b)Use interpolation to upsample.

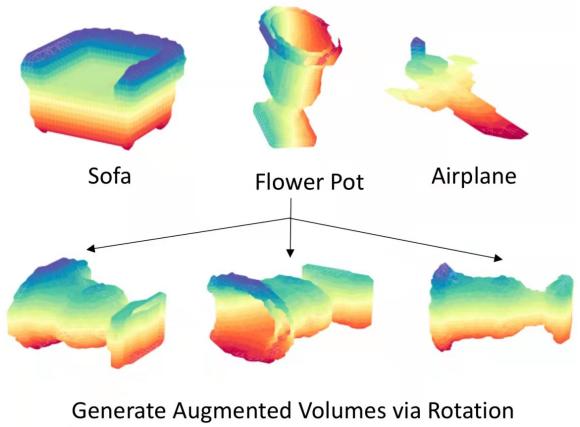

Figure 9: The ModelNet40 dataset consists of several CAD objects from general classes including sofas, flower, pots, and airplanes. These volumes can be augmented using standard image data augmentation techniques such as rotation extended to three dimensions.

Classification on ModelNet40

Table 1: Object classification accuracy on ModelNet40.

Method	Accuracy(%)
PointNet (Qi et al. 2017a)	86.69
PointNet++ (Qi et al. 2017b)	89.81
PointConv (Wu, Qi, and Fuxin 2019)	90.86
Ours	90.67

For a fair comparison, the ModelNet40 datasets for our experiments are preprocessed by (Qi et al. 2017a). By default, 2048 input points are used.Limited to computation ability,all networks are trained with 50 epoches.Table 1 shows the classification accuracy of state-of-the-art methods on point cloud representation. In ModelNet40, our network outperforms PointNet and PointNet++ by 90.67%. Although the original PointConv presents the best result in

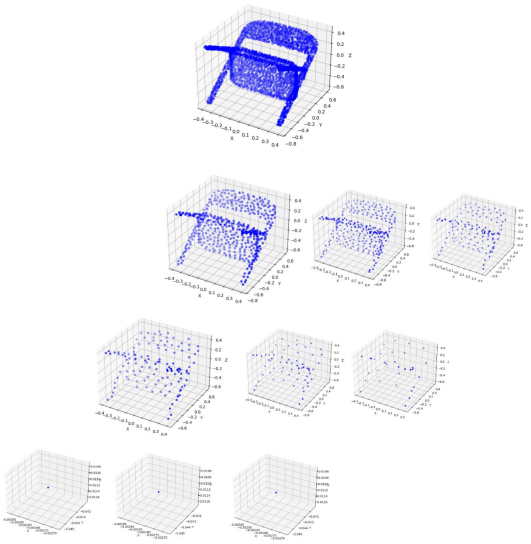

Figure 10: The visualization of MR block.

ModelNet40, its network is trained regardless of density unevenness and not generic to point clouds with non-uniform sampling,while our proposed network is extendable to point cloud classification task with varying density and the local region features obtained by our method are compact and robust.

Effects of MR size

Table 2: Effects of MR size choice on ModelNet40 classification.

MR size	Accuracy(%)
(1024,512,256),(256,128,64),(1,1,1)	90.09
(512,256,128),(128,64,32),(1,1,1)	90.86
(256,128,64),(64,32,16),(1,1,1)	89.44

In this part, we experiment using different MR sizes on the ModelNet40 dataset. For all experiments, three layers of MR block is adopted and the number of input points is 2048. As shown in Table 2, the network with smaller resolution sampling size is more effective and the most beneficial MR block is constructed with size being (256,128,64),(64,32,16),(1,1,1) for each MR layer.

Effects of MR layers

In this part, we mainly discuss the influence of the choice of MR layers on model classification. As with the above experiment, all other variables remain unchanged, mainly changing the number of MR layers. As shown in Table 3, in the three cases of layers in the experiment, the model with an MR layer of 3 performs best.

Table 3: Effects of layers of MR on ModelNet40 classification.

Layers of MR	Accuracy(%)
(512,256,128),(1,1,1)	90.01
(512,256,128),(128,64,32),(1,1,1)	90.86
(512,256,128),(128,64,32),(32,16,8),(1,1,1)	89.98

MR block visualization

At the end of the experiment. We have achieved the visualization of each MR block, based on the model that performed best in the above two test experiments, that is, three MR layers, each with a resolution of (512, 256, 128) (128, 64, 32) (1, 1, 1). As shown in Figure 10, each MR block extracts points and features in the upper layer with different resolutions. For example, the first MR block extracts 512, 256, and 128 points from the original point cloud data set, extracts their local features, and finally transfer the feature fusion to the next layer.

Conclusion

We have proposed a multi-resolutional point cloud classification method named PointConv++. Compared with traditional approaches that are not aware of point cloud density variation, our method takes information interchange among different resolutions into account ,thus enabling robust extraction of point cloud features. Extensive experiments validate the effectiveness and generality of our method.

References

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* .

Hua, B.-S.; Tran, M.-K.; and Yeung, S.-K. 2018. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 984–993.

Jia, X.; De Brabandere, B.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic filter networks. In *Advances in neural information processing systems*, 667–675.

Le, E.-T.; Kokkinos, I.; and Mitra, N. J. 2020. Going Deeper With Lean Point Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9503–9512.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553): 436–444.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. Pointcnn: Convolution on χ-transformed points. In *Advances in neural information processing systems*, 820–830.

Lu, Q.; Chen, C.; Xie, W.; and Luo, Y. 2020. PointNGCNN: Deep convolutional networks on 3D point clouds with neighborhood graph filters. *Computers & Graphics* 86: 42–51.

Masci, J.; Boscaini, D.; Bronstein, M.; and Vandergheynst, P. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, 37–45.

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

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; and Guibas, L. J. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5648–5656.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 5099–5108.

Ravanbakhsh, S.; Schneider, J.; and Poczos, B. 2016. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500* .

Riegler, G.; Osman Ulusoy, A.; and Geiger, A. 2017. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3577–3586.

Savva, M.; Yu, F.; Su, H.; Aono, M.; Chen, B.; Cohen-Or, D.; Deng, W.; Su, H.; Bai, S.; Bai, X.; et al. 2016. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the eurographics workshop on 3D object retrieval*, volume 10.

Shen, Y.; Feng, C.; Yang, Y.; and Tian, D. 2018. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4548–4557.

Simonovsky, M.; and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3693–3702.

Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; and Kautz, J. 2018. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2530–2539.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, 945–953.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38(5): 1–12.

Wu, W.; Qi, Z.; and Fuxin, L. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9621–9630.

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