Report Number:

2019-12-13

Point Cloud Processing with Neural Networks

Stephanie Miller and Jiahao Li

In this project, we explore new techniques and architectures for applying deep neural networks when the input is point cloud data. We first consider applying convolutions on regular pixel and voxel grids, using polynomials of point coordinates and Fourier transforms to get a rich feature representation for all points mapped to the same pixel or voxel. We also apply these ideas to generalize the recently proposed "interpolated convolution", by learning continuous-space kernels as a combination of polynomial and Fourier basis kernels. Experiments on the ModelNet40 dataset demonstrate that our methods have superior performance over the baselines in 3D object... **Read complete abstract on page 2.**

Point Cloud Processing with Neural Networks

Stephanie Miller and Jiahao Li

Complete Abstract:

In this project, we explore new techniques and architectures for applying deep neural networks when the input is point cloud data. We first consider applying convolutions on regular pixel and voxel grids, using polynomials of point coordinates and Fourier transforms to get a rich feature representation for all points mapped to the same pixel or voxel. We also apply these ideas to generalize the recently proposed "interpolated convolution", by learning continuous-space kernels as a combination of polynomial and Fourier basis kernels. Experiments on the ModelNet40 dataset demonstrate that our methods have superior performance over the baselines in 3D object recognition.

Point Cloud Processing with Neural Networks

Jiahao Li
Washington University in St. Louis
jiahao.li@wustl.edu

Abstract

In this project, we explore new techniques and architectures for applying deep neural networks when the input is point cloud data. We first consider applying convolutions on regular pixel and voxel grids, using polynomials of point coordinates and Fourier transforms to get a rich feature representation for all points mapped to the same pixel or voxel. We also apply these ideas to generalize the recently proposed "interpolated convolution", by learning continuous-space kernels as a combination of polynomial and Fourier basis kernels. Experiments on the ModelNet40 dataset demonstrate that our methods have superior performance over the baselines in 3D object recognition.

1 Introduction

Recently, neural networks have gained great success in 2D image understanding [Krizhevsky et al., 2012, He et al., 2016, Ren et al., 2015, He et al., 2017]. The main network architecture used on 2D image data is convolutional neural networks. Convolutional neural networks stack a series of convolutional layers, which aggregate local features hierarchically to extract high-level semantic features. Each convolutional layer applies the same transform to all local regions in the image, thereby extracting information from local spatial structure, while still having a small number of shared learnable parameters.

While 2D images are still the main data source for computer vision tasks, 3D data are becoming easier and easier to collect these days. Equipments for getting 3D data, such as depth cameras and LiDARs, are very popular and become much more affordable than before. 3D computer vision algorithms are now deployed in a wide range of applications, including autonomous driving, robot navigation and facial recognition. 3D data provide richer information about the shapes of objects than 2D images, and thus have the potential to lead to better performance for vision tasks.

Point cloud is a common representation in which 3D data is measured and made available, especially when objects are imaged from multiple views. A point cloud is an unordered set of 3D point coordinates. It is usually stored as an $N \times 3$ matrix, where each row is the (x, y, z) coordinates. Despite its simple form, it is difficult to directly transfer the deep learning methods on 2D images to point cloud data. There are 3 main difficulties for dealing with point cloud data.

- **Unordered data**: Even though point cloud is stored as a matrix, it is intrinsically unordered. Permuting the rows of the point cloud matrix does not change the underlying point cloud itself. As a result, neural network models should be able to achieve some kind of permutation invariance. However, popular neural network structures, such as fully-connected networks and convolutional networks, are not permutation-invariant in nature.
- **Irregularity**: While digital 2D images are rasterized intensity values that are arranged in a regular grid, point cloud data is a set of continuous point coordinates. This poses difficulties to convolution operations since the weights of convolution are stored in a regular kernel.

- **Sparsity**: Points in a point cloud typically only lie on the surface of the object. As a result, they only occupy a very small portion of the 3D space. This property poses challenges to the computational efficiency of neural network models on point cloud data.

In this project, we exploit novel deep learning techniques to deal with point cloud data. In particular, we try to apply convolution on point clouds. We explore two ways of achieving this. We first consider rasterizing the point cloud to convert it into a regular grid—using two different rasterization strategies, and capturing rich features encoding the spatial distribution of points grouped within the same grid element. We also describe a new way of carrying out convolutoins in continuous spatial co-ordinates. In the following sections, we first discuss existing works on 3D point cloud processing, and then describe our methods and innovations.

2 Background

Currently there are 4 main categories of deep learning methods on point clouds.

2.1 Voxel-based methods

A natural way to apply neural networks on point cloud is to first rasterize the point cloud into a regular grid [Allen et al., 2008, Zhou and Tuzel, 2017]. Given a point cloud, we can find a 3D bounding box that contains the point cloud, and divide the bounding box into small cells. These cells are often referred to as "voxels". Due to sparsity and irregularity, each voxel may contain multiple points or no points at all. We then count the number of points in each voxel, and place the value at that voxel. This can be seen as a way of representing the density of the voxel. We can easily apply 3D convolutions on this regular grid to extract features.

There is a trade-off between the resolution of the voxel grid and computational efficiency. If the resolution is high, fine details will be preserved, but the grid would be extremely sparse. Therefore, it is computationally inefficient to directly apply 3D convolution on this sparse grid. If the resolution is low, a lot of details will be lost, and the performance would become worse.

2.2 Image-based methods

Another way to convert a point clouds into regular representation is to render it into 2D images [Su et al., 2015]. A single image is clearly not enough for 3D data. As a result, we put multiple virtual cameras around the object, and render multiple images of multiple views of the same object. The rendered images are then fed into several convolutional neural networks with shared weights indepedently to extract high-level features. Then a pooling layer is used to aggregate the features together to get the final prediction.

Unlike voxel-based methods, we are "compressing" the 3D object into 2D images, so each image is a dense and high-resolution representation. But we are losing resolution in the other dimension, which is the number of views. If we have too many views, it would be computationally inefficient since we need to do multiple forward passes.

2.3 PointNet

PointNet [Qi et al., 2017a] is one of the most successful architectures on point cloud data. It is surprisingly simple but very effective. It first feeds the coordinates of each point in the point cloud to a series of fully-connected layers to extract features indepedently. Then it uses an element-wise max operation to aggregate the features of different points into one feature vector, which is used for final prediction.

A nice property of PointNet is that it is permutation-invariant. Also, it does not rasterize the points so there is no loss of information due to discretization. However, since it extracts features indepedently for each point and then aggregated globally, it loses local information. There is a improved version named PointNet++ [Qi et al., 2017b], which apply small PointNets to local neighborhoods to overcome this difficulty.

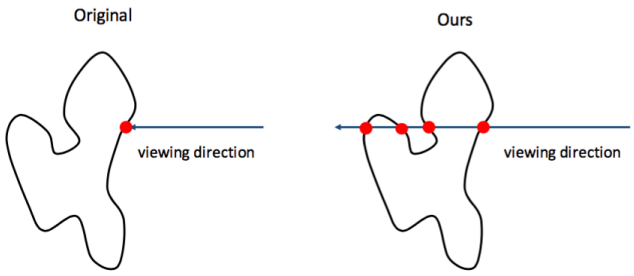

Figure 1: Original image-based method vs ours.

2.4 Graph-based methods

Recently, there are some works that try to apply graph-based methods on point cloud data [Wang et al., 2018]. These works view each point in a point cloud as a vertex in a graph. Edges are drawn between neighboring points. These methods are not particularly related to this project so we do not explain them in detail in this report.

3 Our methods

3.1 Voxel-based method

Original voxel-based methods [Allen et al., 2008] use the number of points as the feature for each voxel. When the grid resolution is low, a lot of fine structures of the point cloud would be lost. We proposed to use a richer representation of the distribution of points in a voxel to overcome this difficulty. For each point in a voxel, we first transform the relative point coordinates to a feature vector. Then we aggregate the features in a voxel by pooling all the feature vectors, using element-wise sum/mean/max operations. In particular, we use polynomial functions and Fourier transform.

Suppose in a voxel, we have a set of points $\{\boldsymbol{p_n}\}$, where $\boldsymbol{p_n} = (x_n, y_n, z_n)$. For the polynomial features, we transform each point in the following way

$$\boldsymbol{q_n} = [1, x_n, y_n, z_n, x_n^2, y_n^2, z_n^2, x_n y_n, x_n z_n, ..., x_n^d y_n^d z_n^d]^\top \tag{1}$$

where $\boldsymbol{q_n}$ is the feature vector for that point, and d is a hyper-parameter that controls the degree of polynomials. Then we aggregate these vectors to get the features for the whole voxel

$$\boldsymbol{f} = \frac{1}{N} \sum_n \boldsymbol{q_n} \tag{2}$$

where N is the number of points in the voxel. This polynomial representation can be seen as an implicit fitting of polynomial surfaces in the voxels, and thus encodes the geometric information of the points in a voxel.

We can also use features in the frequency domain for a voxel. We do Fourier transform on the points in a voxel

$$g(u, w, v) = \sum_n e^{-j2\pi(ux_n + vy_n + wz_n)} \tag{3}$$

where (u, v, w) are the frequencies. We use the transformation for different frequencies as the features for a voxel

$$\boldsymbol{f} = [g(0,0,0), g(1,0,0), g(0,1,0), g(0,0,1), g(2,0,0), ..., g(d,d,d)]^\top \tag{4}$$

where d is a hyper-parameter for controlling the maximal frequencies.

After obtaining the voxel grid containing these features, we feed the grid to a 8-layer 3D convolutional neural network to get the final prediction.

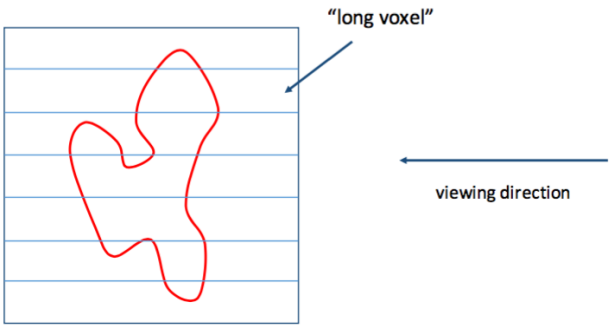

Figure 2: Rasterizing the point cloud into "long voxels".

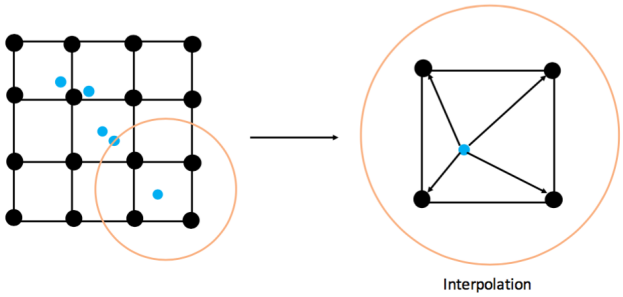

Figure 3: Scattering point features to regular positions.

3.2 Image-based method

We also try to rasterize the point cloud into multiple 2D images instead of a 3D voxel grid. As shown in Figure 1, we aggregate the point features along the view direction, instead of only preserve the depth of the visible point at the front. For simplicity, we set the virtual camera to be infinitely far away, so the viewing directions for all the points are the same. This way, we can divide the grid into "long voxels" along the given direction (Figure 2). Then we can use the same kind of polynomial and Fourier functions described in the previous section to get an aggregated feature vector.

We then use a similar architecture as MVCNN [Su et al., 2015], but only use 3 different views for efficiency.

3.3 Convolution on continous space

There is a recent paper [Mao et al., 2019] that proposes "interpolated convolution", which is a convolution operation defined on continous space. For each point, we crop a cubic neighborhood from point cloud. Then we place a regular kernel on the neighborhood, and scatter the features of each point in the kernel to the regular kernel points by trilinear interpolation (Figure 3). Then we can do matrix multiplication at each kernel point of the weight matrix and the feature vector.

Note that this "scattering" operation is essentially the same as "gathering" weight matrices from the kernel points to the points in the kernel. Mathematically, this can be written as

$$Y(p) = (X * W)(p) = \frac{1}{N_p} \sum_{p'} X(p') \cdot W(p - p') \tag{5}$$

where p is a center point where the kernel is placed, p' is a point in the neighborhood. $X(p)$ is the input features and $Y(p)$ is the output features. $W(x)$ is a continous weight function defined on the 3D space. N_p is the number of points in the neighborhood.

Table 1: Performance of our voxel-based methods on the ModelNet40 dataset.

Model	Grid Size	Accuracy (%)
Count	8	83.0
Polynomial	8	89.9
Fourier	8	88.0
Count	16	84.8
Polynomial	16	90.2
Fourier	16	90.5

Table 2: Performance of our image-based methods on the ModelNet40 dataset.

Model	Accuracy (%)
Min	88.3
Mean	84.8
Polynomial	90.6
Fourier	91.1

We can use a sum of basis functions for $W(x)$. Suppose we have a set of weight matrices $\{W_i\}$, we can write it as

$$Y(p) = (X * W)(p) = \frac{1}{N_p} \sum_{p'} X(p') \sum_i T(p' - p, q_i) W_i \tag{6}$$

where $T(\cdot, \cdot)$ is a trilinear interpolation function, and q_i is the position of some regular kernel point.

The trilinear interpolation function is a piecewise linear function. Our idea is that we can replace it with some more complicated basis functions, such as polynomial basis functions. We can write the above equation as

$$Y(p) = (X * W)(p) = \frac{1}{N_p} \sum_{p'} X(p') \sum_i T_i(p' - p) W_i \tag{7}$$

where $T_i(p' - p)$ is not associated with some kernel point position q_i. We tried replacing it with polynomial functions of the form

$$T_i(p) = x^{d_{i1}} y^{d_{i2}} z^{d_{i3}} \tag{8}$$

Also, we tried out Fourier basis functions

$$T_i(p) = e^{-j2\pi(u_i x + v_i y + w_i z)} \tag{9}$$

4 Experiments

4.1 Dataset

We evaluate our model with the 3D object recognition task on the ModelNet40 dataset [Wu et al., 2014]. ModelNet40 is a 3D shape dataset consisting of shapes from 40 different categories. There are 9843 shapes in the training set and 2468 shapes in the test set. We sample 1024 points from each shape for training and inference. To avoid overfitting on the test set, we use 80% examples in the training set for training, and the other 20% for validation.

4.2 Results

Table 1 shows the results of voxel-based methods evaluated on the ModelNet40 dataset. We can see that both the polynomial and Fourier features outperform the "count" baseline by a large margin. Also, increasing the resolution of the voxel grid can further boost the performance of all the models. Note that polynomial and Fourier features with a grid size 8 still perform much better than the "count" feature model with grid size 16. This shows that these rich feature representations effectively encode the geometric information inside voxels.

Table 3: Performance of generalized interpolated convolution on the ModelNet40 dataset.

Model	Accuracy (%)
Trilinear Interpolation	88.9
Polynomial	89.8
Fourier	87.8

Table 2 shows the results of our image-based methods. The baselines here are "Min" and "Mean", which mean taking the min and mean of values inside a voxel. Still, our methods outperform the baselines. Also, the image-based methods give overall better results than those for the voxel-based methods. This may be due to the larger and deeper 2D convolutional networks applied on them.

Table 3 shows the results of our generalized interpolated convolution. While the polynomial basis functions outperform the trilinear interpolation baseline, Fourier functions are not very effective. Maybe a careful choice of frequencies would help improve the results.

References

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

Michael Allen, Lewis Girod, Ryan Newton, Samuel Madden, Daniel T. Blumstein, and Deborah Estrin. Voxnet: An interactive, rapidly-deployable acoustic monitoring platform. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 371–382, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3157-1. doi: 10.1109/IPSN.2008.45. URL https://doi.org/10.1109/IPSN.2008.45.

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017. URL http://dblp.uni-trier.de/db/journals/corr/corr1711.html#abs-1711-06396.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5099–5108. Curran Associates, Inc., 2017b. URL http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.pdf.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. URL http://arxiv.org/abs/1801.07829.

Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014. URL `http://arxiv.org/abs/1406.5670`.