Neural Network for 3D object classification

Lin Shao
ICME
Stanford University, 94305, CA
lins2@stanford.edu

Peng Xu
ICME
Stanford University, 94305, CA
pengxu@stanford.edu

Abstract

3D object classification is an interesting topic especially when large scale 3D CAD datasets are available. A convolutional neural network combining spatial transformation network is used to classify 3D objects in a subset of ModelNet. The spatial transformation network is an attempt to deal with rotation invariance problem in 3D object classification. We evaluate our method by comparing with previous result.

1. Introduction

Recent years convolutional neural network has gained great successes on 2D image classification [2] detection and locations. An interesting question following is can convolutional network be applied to 3D model classification? With more and more 3D CAD model data sets available, we are now exploring this question. There are several large 3D datasets. Trimble 3D warehouse contains 2.5M models in total,Yobi3D has 1M models.

In this project, we focus on the classification of 3D model problem. The model we use are the 40-classes subset of ModelNet [6] which is a large scale 3D CAD Dataset. The method of representation 3D model is voxelization within a cube with 32 x 32 x 32. Every voxel takes value between 0 and 1. 1 indicates the occupation by the 3D model in the small voxel while 0 indicates the voxel is empty. We are trying to construct our convolutional neural network based on this voxelization.

When the classification object expand from 2D to 3D, transformation capacity of the object get larger because the rotation could be done in 3D spaces. All the affine transformations make the classification complicated. In this cases, simple conducting data augmentation and putting all the models into the neural network to train the model become not so applicable. So one of our focus is to deal with translation and rotation invariance problem.

To solve translation and rotation invariance problem, we adopted the idea from Jaderberg et.al [1] to use a spatial transformer layer to have the effect of alignment.

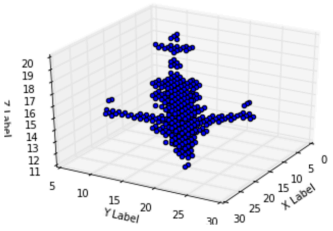

Figure 1: Voxelization example of airplane instance.

2. Problem Satement

In this project, we want to solve 3D object classification on the 3D domain.

Dataset: ModelNets ModelNets dataset was introduced by [6]. There are two ModelNets datasets: `ModelNets10` and `ModelNets40`. `ModelNets40` contains 12,311 shapes covering 40 common categories. Categories of 40 classes are airplane, bathtub, bed, bench, bookshelf, bottle, bowl, car, chair, cone, cup, curtain, desk, door, dresser, flower pot, glass box, guitar, keyboard, lamp, laptop, mantel, monitor, night stand, person, piano, plant, radio, range hood, sink, sofa, stairs, stool, table, tent, toilet, tv stand, vase, wardrobe, xbox. And `ModelNets10` is a subset of `ModelNets40`. For our experiments, we use the same training and test split as [6] provided.

Basically there are two problems we attempt to address:

1. What is the best representation of 3D models? In this paper, we focus on the voxelization representation. The simplest voxelization representation is to set 0-1 binary value to each voxel (as shown in Figure 1). We take this as a baseline. It is worthy to try more complex representations.

2. What is the best/good architecture of deep learning models to train 3D voxel data? A reasonable architecture is to extend the 2D convolutional networks, i.e. the classic structure to train image data, to a 3D convolutional networks, as this is done by [3]. So we take their model as our baseline and investigate other architectures.

3. Technical Approach

Baseline We take 0-1 binary voxel representation of the all 3D models (Figure 1). We augment the data by rotating every single model every 30° along the gravity direction. And we test the VoxNet [3] as our baseline model. The architecture of VoxNet is shown in Table 1.

Layers	Parameters
fully connect	40
drop3	p = 0.4
fully connect	128*10
drop2	p = 0.4
pool2	pool shape [2 2 2]
conv2	receptive field 3x3x3 filterNum 32
drop1	p = 0.2
Conv1	receptive field 5x5x5 filterNum 32
Input Layer	Size 32 x 32 x32

Table 1: Architecture of VoxNet.

3.1. Complex representation

As we mentioned in the previous section, we first will try more complex representations than just 0-1 binary voxel values. Under the same voxelization, for each voxel, we assign the squared minimum distance to the surface of the 3D object. In the representation, all the voxels in the object have value 0. There are two empirical benefits here. First, this solves the sparsity issue of the simple 0-1 binary representation; Second, intuitively the distribution of the voxel values over the whole voxel cubic is approximately rotation invariant because the distance is independent of the absolute position of the voxel but relied on the relative distance to the object itself.

3.2. 3D spatial transform network

Recently [1] proposed a spatial transform network to deal with this problem. The basic idea is to consider the spatial transform as regression problem as we can specify the transformation parameters. The architecture is shown in Figure 2. We adopt this idea, and built a 3D spatial transform network into the VoxNet. The general architecture is in Figure 3.

Basically, 3D-SPN contains 3 parts. The first part is a localization net, in which the goal is to find the best 3D affine transformation to predict the right labels, specifically we take the same architecture of VoxNet, except the output layer has only 6 neurons which are the parameters θ of the affine transformation. The architecture of the location net is shown in Table 2. The second and third parts of 3D-SPN is Grid generator and feature sampler. Basically these two parts is to apply 3D affine transformation which is set by localization net to the original data to get transformed voxel data. The following[1] shows more details about 3D-SPN.

Localization network The localization networks takes the input feature map $U \in \mathbb{R}^{H\times W\times L}$, where $H \times W \times L$ is the size of the voxel data and outputs θ, the parameters of the transformation $\mathcal{T}_\theta$ to be applied to the feature map: $\theta = f_{loc}(U)$. In general f_{loc} can take any architecture that we can specify. Here in our model, we use the same convolutional network structure in VoxNet.

Parameterised Sampling Grid In order to get the right output feature V after certain transformation $\mathcal{T}_\theta$, we need parameterize the sampling grid of the input U. Specifically, for each output voxel (x_i^t, y_i^t, z_i^t) must be on a regular grid G. Here we consider a 3D affine transformation in (1). Then given $\mathcal{T}_\theta$, we can find the position of the source voxels (x_i^s, y_i^s, z_i^s) by (1).

$$\begin{pmatrix} x_i^s \\ y_i^s \\ z_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{pmatrix} \theta_1 & \theta_{12} & \theta_{13} & \theta_{14} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{34} \end{pmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ z_i^t \\ 1 \end{pmatrix} \quad (1)$$

where (x_i^t, y_i^t, z_i^t) is the target position of the output voxel data and (x_i^s, y_i^s, z_i^s) is the source position of the input voxel data. And we used normalized coordinates such that $-1 \leq x_i^t, y_i^t, z_i^t \leq 1$ when within the spatial bounds of the output and $-1 \leq x_i^s, y_i^s, z_i^s \leq 1$ when within the spatial bounds of the input. In our model, currently we only consider the transformations along the gravity direction so we constrain the transformation to be

$$\mathcal{T}_\theta = \begin{pmatrix} \theta_1 & \theta_{12} & 0 & \theta_{14} \\ \theta_{21} & \theta_{22} & 0 & \theta_{24} \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Differentiable Image Sampling After finding the right correspondence of positions between input U and V, we need fill in the voxel values in the output. Specifically, for each target position (x_i^t, y_i^t, z_i^t) , the voxel value V_i is defined as the average at (x_i^s, y_i^s, z_i^s) in the input U by some

[1] Most the contents here is from [1], but we extend it to 3D settings.

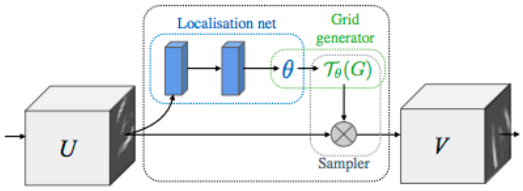

Figure 2: spatial transformation network. Basically, SPN constain 3 parts. the first part is a localization net, which is doing regression over the parameter θ of the affine transformation. The second part is Grid generator, which finds the exact position of the voxel after the affine transformation. The third part is Sampler of image features.

sampling kernel as follows:

$$V_i = \sum_n^H \sum_m^W \sum_l^L U_{nml} k(x_i^s - m; \Psi_x) k(y_i^s - n; \Psi_y) k(z_i^s - l; \Psi_z) \tag{2}$$

where V_i is the voxel value at (x_i^t, y_i^t, z_i^t), Ψ_x, Ψ_y and Ψ_z are the parameters of a generic sampling kernel $k()$. Specifically we use a bilinear sampling kernel, which gives

$$V_i = \sum_n^H \sum_m^W \sum_l^L U_{nml} \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|). \tag{3}$$

In this scheme, we can derive the backpropagation of the loss by computing the gradients:

$$\frac{\partial V_i}{\partial U_{nml}} = \sum_n^H \sum_m^W \sum_l^L \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|)$$

$$\frac{\partial V_i}{\partial x_i^s} = \sum_n^H \sum_m^W \sum_l^L \left[\max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|) \cdot \begin{cases} 0 & |m - x_i^s| \geq 1 \\ 1 & x_i^s \leq m < x_i^s + 1 \\ -1 & x_i^s - 1 < m \leq x_i^s \end{cases} \right].$$

4. Experiment Results

experiment setting We test our model in `ModelNets40`. In the training set of `ModelNets40`, we augmented the data by rotating each model 30° along the gravity direction. In the test set of `ModelNets40`, we have two sets of test data. One set is rotating every single model 30° along the gravity direction as in the training set. The other set is rotating every one 12° along the gravity direction.

Layers	Parameters
fully connect	6
drop3	p = 0.4
fully connect	128*10
drop2	p = 0.4
pool2	pool shape [2 2 2]
conv2	receptive field 3x3x3 filterNum 32
drop1	p = 0.2
Conv1	receptive field 5x5x5 filterNum 32
Input Layer	Size 32 x 32 x32

Table 2: Architecture of localization net in 3D-SPN.

Figure 3: Architecture of 3D-SPN-VoxNet. The input is the 3D voxel data, and first go through 3D-SPN, then feed into VoxNet and get the output.

First we show the benchmark results as in Table 3[2].

Algorithm	ModelNet40	ModelNet 10
MVCNN[5]	90.1%	NA
VoxNet[3]	86.22%	92%
DeepPano[4]	77.63%	85.45 %
3DShapeNets[6]	77.3%	83.5 %

Table 3: Classification accuracy benchmark on ModelNets

4.1. Complex Representation

In this section, we test the idea in Section 3.1. The results are in Table 4. We found that have 1.2% improvement on the test accuracy comparing using 0-1 binary data trained on VoxNet.

4.2. 3D spatial Transformer Network

In this section, we combine the 3D spatial Transform Network with VoxNet, which we call 3D-SPN-VoxNet.

[2]In our experiments, we obtained 86.22% accuracy on `ModelNet40` using VoxNet as 83% accuracy reported in [3].

Algorithm	ModelNet40-12	ModelNet40-30
Dist+VoxNet	**87.44%**	NA
01+VoxNet	86.22%	85.45%

Table 4: Classification accuracy using data of different representations. “Dist+Voxnet” means using the distance based voxel data to train on VoxNet. “01+VoxNet” means using the standard 0-1 binary voxel data to train on VoxNet.

Here we use the standard binary voxel data to train on those models. The results are in Table 5. We can see that we achieve 2% improvement on the test accuracy.

Algorithm	ModelNet40-12	ModelNet40-30
3D-SPN-VoxNet	**88.25%**	**86.39%**
VoxNet	86.22%	85.45%

Table 5: Classification accuracy.

4.3. Visualization

We extract the features of 3D SPN which are new voxelizations. We then compare new voxelizations with the origin voxelizations to understand how the 3D SPN works.

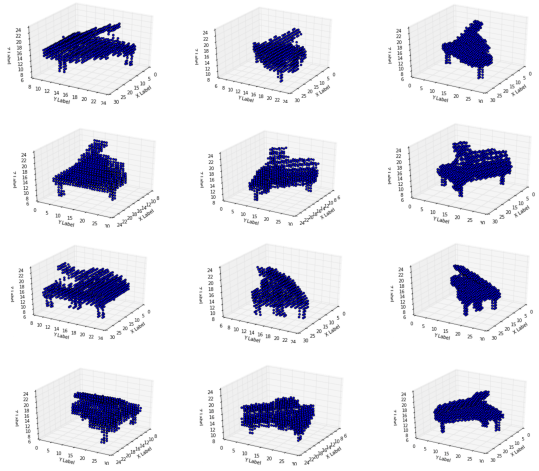

Figure 4: Input Voxelization of Piano

Figure 4 shows an instance of piano. The voxeliazation is rotated every 30 degrees starting from left to right then from top to down. There voxelizations are the input of our 3D-SPN-VoxNet model. Figure 5 shows the results of SPN. The twelves voxelizations correspond to the voxelizations in Figure 4 one by one. After SPN, the voxelizations are aligned along one direction of cube surface. It indicates that SPN actually has the effect of alignment.

Figure 5: Output Voxelization of Piano after spatial transform layer

Figure 6 and 7 shows the input and output voxelization of a bed model. The output voxelization of SPN also get aligned along one direction of cube surface. Compare the alignment direction of bed with the alignment direction of piano. In these different alignments, the differences between the piano and bed get larger while the shape differences within the same models get minimized.

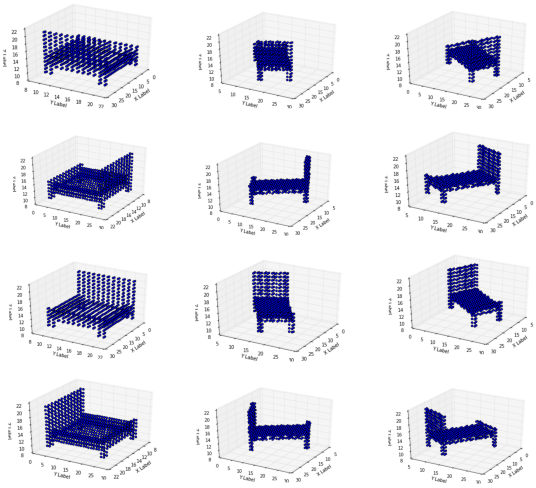

Figure 6: Input Voxelization of Bed

However the effect of alignment does not work for all voxelizations. Figure 8 and 9 show the visualization of airplane categories. Voxelizations in Figure 8 are the origin

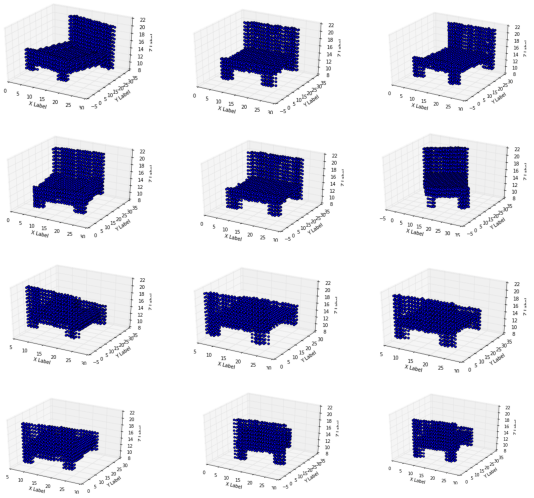

Figure 7: Output Voxelization of Bed after spatial transform layer

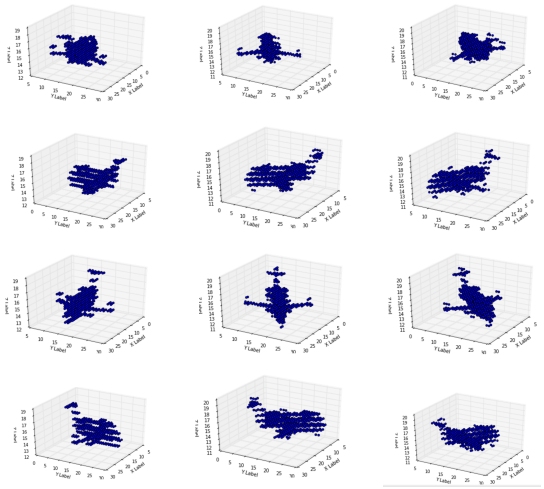

Figure 8: Input Voxelization of Airplane

inputs. Voxelizations in Figure 9 are the output of SPN. There is no obvious alignment effect of SPN since the output voxelizations in Figure 9 also rotate.

It could be explained why SPN has no alignment effect on voxelizations in airplane category. The origin Voxnet classification accuracy in airplane category has already achieved 100 percents. It indicates that airplane voxelizations are easily recognized by a classical CNN models. No matter what kind of degrees the airplane voxelizations rotate, the classical CNN models could recognized it. Then the loss function in the final layer will get zero scores. There is no back propagation to update the parameters in the SPN.

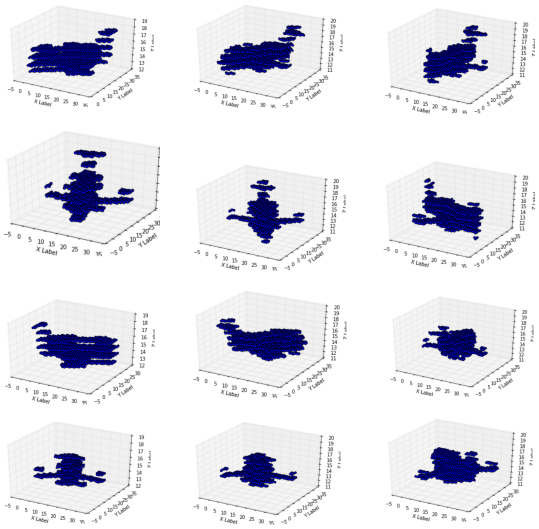

Figure 9: Output Voxelization of Airplane after spatial transform layer

5. Discussion

By visualizing we realize that the SPN architecture has the effect of alignment. Models are intended to align along surface directions of the cube. Based on the fact, we realize that there is an important assumption. There are some special directions that if models are aligned along these directions then the total differences between 40 categories models get maximized.

When models are aligned the shape variation within a same category is much smaller right now. Without SPN, data augmentation will be conducted. Same models will rotate several times to cover all possible rotations. This type of augmentation will not work if we allow the rotation axis to change in 3D space. The CNN models are required to be more complicated to classify so many instances of models. It will become difficult to train such a neural network.

An interesting and important question following is why the most models are aligned along these special directions of the cube. Current we could not give a reasonable explanation.

Besides rotation, the SPN also have the effect of zooming. The output voxelization becomes larger than the input voxelization which means there are more points to represent the model.

Based on the alignment and zooming effect of SPN, we could redesign more complicated CNN models after the SPN architectures. Because the input of later CNN models are aligned and adjusted their sizes to be better recognized.

References

[1] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. 2015.

[2] A. Krizhevsky, I. Sutskever, and G. 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.

[3] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015.

[4] B. Shi, S. Bai, Z. Zhou, and X. Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *Signal Processing Letters, IEEE*, 22(12):2339–2343, 2015.

[5] 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*, pages 945–953, 2015.

[6] A. K. F. Y. L. Z. X. T. J. X. Z. Wu, S. Song. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015.