MORE FOR LESS: INSIGHTS INTO CONVOLUTIONAL NETS FOR 3D POINT CLOUD RECOGNITION

Usama Shafiq* Murtaza Taj* Mohsen Ali†

*LUMS Syed Babar Ali School of Science and Engineering, †Information Technology University

ABSTRACT

With the recent breakthrough in commodity 3D imaging solutions such as depth sensing, photogrammetry, stereoscopic vision and structured light, 3D shape recognition is becoming an increasingly important problem. A longstanding question is what should be the format of the 3D shape (such as voxel, mesh, point-cloud etc.) and what could be a good generic feature representation for shape recognition. This question is particularly important in the context of convolutional neural network (CNN) whose efficacy and complexity depends upon the choice of input shape format and the design of network. It has been seen that both 3D voxel representation as well as collection of rendered views on 2D images have produced competing results. Similarly, it have been seen that networks with few million parameters and networks with several hundred million parameters have similar performance.

In this work we compare these solutions and provide an analysis on the factors resulting in increase in the parameters without significantly improving accuracy. On the basis of the above analysis we propose a representation method (point cloud to 2D grid) and architecture that results in much less parameters for the CNN but has competing accuracy.

Index Terms— Point cloud, 3DOR, recognition, deep learning

1. INTRODUCTION

Due to technological advancement in the last decade, 3D data has become readily available in the form of point cloud obtained from commodity depth sensors such as Kinect, Infineon REAL3^{TM}Image Sensor and LiDAR. On basis of the resounding success of Convolutional Neural Network (CNN) for the image classification and object detection, CNN has also been employed to solve the task of object recognition in the 3D domain. For the training of CNN, most important questions are about its architecture, that depends upon type of input data (2D, 2.5D, 3D), number of filters and their dimensions, and overall number of parameters that needs to be optimized during training. Larger the number of parameters more difficult it is to train a CNN.

Different object recognition approaches use different transformations of the 3D data i.e. 2D images obtained by projecting point cloud onto 2D plane, depth images, CAD models, volumetric representations, and point clouds themselves. 3D shape formats, such as point cloud, contains inconsistent number of samples and are first projected onto a regular 2D or 3D grid before further processing. Due to 3D nature of the problem and depending upon the representation used, both 2D and 3D convolution have been employed.

Existing literature mainly focuses on two aspects of the problem i) what shape format to be used for the 3D object and ii) which network configuration can best learn its representation. In the case of 3D shape format there have been two popular choices a voxel representation [1, 2, 3, 4, 5] and a collection of rendered views in 2D images [6, 7, 8, 9]. The only exception being FusionNet [10] that aggregates the output obtained from both these input formats.

Although different CNN based approaches have much varying number of parameters, in many cases, the 2D and 3D convolution based approaches achieved similar accuracy [1, 10, 6, 8, 9, 2](see Table 1 for detailed comparison). For example, FusionNet [10] used 3D voxel view approach with 118M parameters and obtained 93.11% accuracy whereas a multi-view approach based on image pairs [6] used 2D projection-based approach having 205.7M parameters and obtained 92.8% accuracy.

Among these comparable approaches, 3D voxel of resolution $32 \times 32 \times 32$ and multiple 2D images of 224×224 have been popularly used as an input format. Both of these input format result in having over $30,000$ input neurons. We instead propose a new unified input format which combines the advantages of both voxel representation as well as 2D image sequences while having the minimum number of input nodes as compared to the existing literature. Our input format is based on a $10 \times 10 \times 10$ voxel representation which is then converted into a 10×10 density image thus resulting in only 100 neurons at the input layer. We show that this $10 \times 10 \times 10$ voxel representation contain least amount of redundancy as compared to higher resolution voxel and provides its empirical as well as statistical evidence.

Furthermore, we also provide a comparative of existing 3D object recognition approaches based on four main factors namely input size, type of convolution (2D or 3D), number of filters and the resulting parameters (see Table 1). We provide analysis of the amount of information encoded in various

Table 1. Comparison between various architectures showing the effect on convolution type, input format and number of filters on the resulting accuracy and training parameters.

Algorithm	ModelNet10	Conv.	No. Params	Input Size	No. of filters
3D-GAN [5]	91.00%	3D	23,086,994	64 x 64 x 64	3280
VRN Ensemble [4]	97.14%	3D	18,000,000	32 x 32 x 32	992
ORION [3]	93.80%	3D	$>$ 921,736	32 x 32 x 32	64
FusionNet [10]	93.11%	2D & 3D	118,000,000	30 x 30 x30	2230
Pairwise [6]	92.80%	2D	205,794,880	224 x 224	3776
GIFT [8]	92.35%	2D	-	224 x 224	-
VoxNet [1]	92%	3D	921,736	32 x 32 x 32	64
DeepPano [9]	85.45%	2D	-	64 x 160	1248
3DShapenet [2]	83.50%	3D	8,812,880	30 x 30 x 30	720
Proposed Architecture	93.06%	2D	1,385,500	10 x 10	500

layers of the network and our proposed approach reduces the redundant and less informative components. The proposed approach uses 2D convolution and it has much less number of parameters as compared to most of the previous architectures yet it achieves a comparable accuracy of 93.06%.

2. METHODOLOGY

Deep Neural Network (DNN) consists of the input layer, output layer and multiple hidden layers between them. One of the main property of the DNN is that it learns the details of the object at different abstraction layers, the initial layers capture simple features where as higher ones capture more complex information. Therefore number of parameters depends upon how many hidden layers are there; on the other hand the fully connected layers are most dense and have huge number of parameters; these parameters are dependent upon the subsampling layers (pooling layers) and the most importantly the inputs size. The input size and dimensions have a vital impact on the overall depth of the network and its parameters. Larger the number of the parameters more difficult it will be to generalize the network when trained on small number of examples.

In order to design a network architecture which requires less number of training parameters without compromising accuracy we focus on two main aspects of the network i) type and size of input and ii) number and size of filters. Next section discusses these two aspects and the proposed architecture on the basis of these insights is shown in Fig. 4.

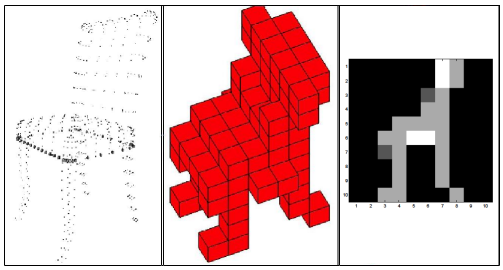
2.1. Input Layer

One of challenges in working with the point cloud is that each object contains different number of points whereas most learning algorithms requires fixed length feature vector as its input. Different techniques are used to convert the point cloud objects into format where each sample is of same size. A simple approach is to uniformly sample the points and use their X, Y, Z co-ordinates as sequential data like RGB values of an image. However, giving co-ordinates as input vector does not exploit neighborhood relationship and thus results in poor performance.

A commonly used alternate is to represent the 3D data as one or more 2D images obtained by projecting point cloud onto 2D plane(s). Shi et al. [9] proposed a DeepPano method that used cylinder projection technique for this purpose and obtained a panoramic view of the 3D object. Multiple 2D views of 3D shape model was used by Su et al. [7], these images were obtained by rotating the 3D model by 30 times. Although 2D projects result in information loss however these approaches still produce competing results.

One of the popular input format for 3D data is voxel [1]. The 3D space is divided into regular-grid and value in each cell indicates occupancy of the cell by the 3D point data. Voxel in essence is similar to pixel in an image. Mostly, a 1-bit representation is used for storing voxel values as there is a negligible difference in results of binary voxel and density voxel [3]. Therefore, we use binary voxels in experiments to save memory. Figure 1 shows the result of voxelization.

A brief comparison of various DNN architectures proposed for 3DOR is presented in Table 1. The input layer used with each of these architectures and its implication on the DNN and its performance can be understood from this table. These techniques used different voxel sizes and hence have different number of training parameters. For example, number of parameters for $64 \times 64 \times 64$ voxel input is twice of number of parameters for $32 \times 32 \times 32$ voxel input. Now the question arises why different input sizes gives similar accuracy. We observed that due to the quantized nature of the binary



(a) Vertices (b) Voxel (c) Density

Fig. 1. Example showing input vertices of a chair, its voxel representation and the resulting density image.

voxel, decreasing the size of voxel input between $64 \times 64 \times 64$ to $10 \times 10 \times 10$ the loss in information is insignificant. Figure 2 shows the results of different voxel sizes and their slices. It can be seen higher resolution voxels contain lot of redundant data, that is, many neighboring voxel slices capture same shape information (Fig. 2). Thus decreasing voxel size mainly decreases the redundant information. The drawback of using voxel is the computational cost of 3D convolution, which increases cubically with the size of 3D voxel view. To overcome computational challenges we instead propose a new data representation called *density image*.

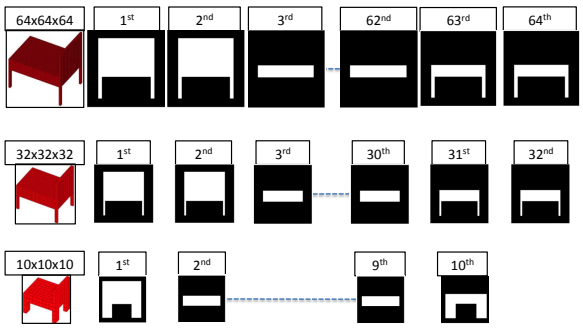


Fig. 2. Comparison between different voxel sizes and their slices showing redundant information among multiple inner slices.

2.2. Density Image

Unlike existing approaches, which relied on the binary voxels, we use a representation that captures similarity of data across the voxel slices. Our motivation could be seen in Figs. 3 and 2.

Figure 3 show a voxel for a circular and a square table, however their 2D slices appear very similar. This indicates that by decreasing the size of the voxel some of the information is indeed lost. However, this decreases the interclass variation between training samples and provides generalization thus resulting in a better training accuracy which is also evident from the experiments shown in Table 2.

Figure 2 (row 2) shows that even in a low-resolution $10 \times$

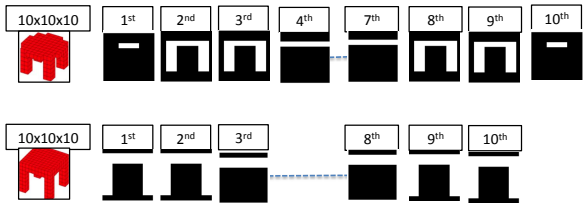


Fig. 3. Comparison between a circular table (see row 1) and a square table (see row 2) showing that they both appear very similar in a $10 \times 10 \times 10$ voxel.

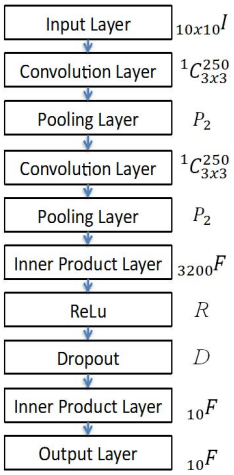
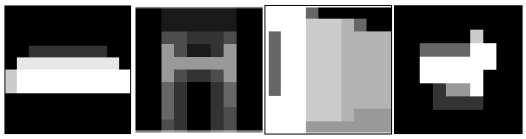


Fig. 4. Proposed deep learning architecture.

10×10 voxel 1^{st} and 2^{nd} slice are exactly same , similarly 3^{rd} - 8^{th} slice are all same. This similarity in slices indicates the length of the particular attribute of the 3D object. To encode this information in our input layer we proposed a new input format called density image, obtained by taking a mean of all the slices (see Fig. 5). We used unnormalized density image as it is challenging to train a DNN on very small numeric values. Using this density image the length of our input layer is further reduced to 100 nodes only.

2.3. Convolutional Layer

The convolution layer is used to exploit the spatial relationship between the neighboring pixels of the voxel input. The spatial features, which are calculated using convolution layer, forms a hierarchy of the complex features [11] and are used for classification. In order to limit the number of parameters, it can be seen from Table 1 that most approaches use less number of filters in initial layers and this number gradually increases in higher low-resolution layers. This may result in losing important feature information and which may result in learning poor representation. An interesting observation is that having a very low-resolution 10×10 input layer as in our case, even initial layers can use large number of filters. This is because convolution with input of smaller spatial size, will result in output of even smaller size and will not have



(a) Bed (b) Chair (c) Monitor (d) Sofa

Fig. 5. Density images / normalized mean of different voxel inputs.

significant effect on the overall number of parameters, however it can help in achieving higher accuracy. For e.g. in the proposed architecture we used 250 filters in the first convolutional layer.

In general, final convolutional layers uses large number of filters whose output is then used to form fully connected layer. In our case we kept the number of filters to 250 in the final convolutional layer as well thus resulting in 500 filters in total. The convolution kernel used in both these layers is of size 3×3 with stride 1.

2.4. Inner product, ReLU and Dropout layers

The other layers of the proposed architecture include inner product, ReLU and dropout layers. The inner product or fully connected layer is used to get the high level understanding of all the data that is being processed in previous layers. ReLU layer is used to optimize the non-linear properties of decision function. It also makes learning process faster without any loss of accuracy of the training model.

Due to the difficulty of labeling 3D dataset, our training data was quite small and this caused over-fitting problem. To solve the over-fitting problem, dropout layer is used. The threshold of 0.75 is used as a dropout probability to keep or drop the input. Figure 4 shows the proposed deep learning architecture.

3. RESULTS

We performed the comparison and evaluation of the proposed architecture on commonly used MODELNET10 [2] dataset. It consist of 10 classes with 4901 samples of 3D CAD models from everyday objects. We used $3,992$ samples for training and 909 samples in test set. For each experiment, during training we used a batch size of 100 samples and performed at least 5000 iterations with varying learning rate.

3.1. Experiment 1: Varying input size

Table 2 shows the effect of changing the size of our density image on the accuracy. We tested our network on five different input sizes and trained using three different learning rates eta. It can be noticed from Table 2 that in 9 out of 12 cases the accuracy increases by decreasing the input size. The best accuracy was achieved at $eta = 0.001$ for which at input size $30 \times 30 \times 30$ the accuracy is 83% whereas for low-resolution input size $10 \times 10 \times 10$ the accuracy is 93%.

Table 2. Experiment 1 showing classification Score in $\%$ with different learning rates (η) and input size.

η	Input Size				
	10×10	15×15	20×20	25×25	30×30
0.001	**93**	90	90	86	83
0.0001	88	84	89	84	81
0.00001	71	81	85	83	82

3.2. Experiment 2: Varying Number of filters

The purpose of this experiment is to understand the effect of increasing the number of filters. It is clear from the Table 3 that the accuracy increases by increasing the number of filters. With only 64 filters accuracy is 73% whereas with 500 filters accuracy increased to 93.06% after which it decreases due to lack of training samples. This makes our approach comparable with Orion [10] and FusionNet [3] that has slightly higher accuracy of 93.30 and 93.11 respectively. However, it can be seen from Table 1 that FusionNet achieved this result with $118,000,000$ parameters as compared to only $1,385,500$ parameters in our case i.e. 85 times less parameters. This is also the lowest number of parameters among the competing approaches ($\approx> 93\%$) for which the number of parameters were publicly available.

Table 3. Accuracy of our CNN architecture with different number of filters, on ModelNet10 dataset.

Filters	**64**	**200**	**400**	**500**	**600**
Accuracy	73	80.00	88.33	**93.06**	86.76
Parameters	114,560	419,200	1,018,400	1,385,500	1,797,600

4. CONCLUSIONS

The paper shows that both the input size and the number of filters significantly impact the number of training parameters as well as the accuracy of a DNN. However, having high-resolution input, complex network and more parameters does not guarantee optimal performance. We present an evaluation and analysis of various design choices for CNN and identified redundant components. Based on this analysis we propose a CNN design, for 3D point cloud recognition, whose each layer is critically designed to achieve more with less parameters. We also present a novel way of representing 3D point cloud data, that is first performing low resolution voxelization and then converting it into 2D density map. The achieved accuracy is 93.06% with 85 times less number of parameters as compared to previous architectures.

5. REFERENCES

[1] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Hamburg, Germany, Sep 2015, pp. 922–928.

[2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Boston, MA, USA, Jun 2015, pp. 1912–1920.

[3] N. S. Alvar, M. R. Zolfaghari, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," CoRR, vol. abs/1604.03351, Apr 2016.

[4] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," CoRR, vol. abs/1608.04236, Apr 2016.

[5] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in Neural Information Processing Systems, Barcelona, ES, Dec 2016.

[6] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, Jun 2016, pp. 3813–3822.

[7] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in IEEE International Conference on Computer Vision, ICCV, Santiago, Chile, Dec 2015, pp. 945–953.

[8] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki, "GIFT: A real-time and scalable 3D shape search engine," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, Jun 2016, pp. 5023–5032.

[9] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3D shape recognition," IEEE Signal Process. Letters, vol. 22, no. 12, pp. 2339–2343, Dec 2015.

[10] V. Hegde and R. Zadeh, "Fusionnet: 3D object classification using multiple data representations," CoRR, vol. abs/1607.05695, 2016.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European Conference on Computer vision, ECCV, Zurich, CH, Sep 2014, Springer, pp. 818–833.