Large-Scale Shape Retrieval with Sparse 3D Convolutional Neural Networks

Alexandr Notchenko[12], Ermek Kapushev[12], and Evgeny Burnaev[1]

[1] Skolkovo Institute of Science and Technology,
{alexandr.notchenko, ermek.kapushev}@skolkovotech.ru
e.burnaev@skoltech.ru
[2] Institute of Information Transmission Problems

Abstract. In this paper we present results of performance evaluation of S3DCNN — a Sparse 3D Convolutional Neural Network — on a large-scale 3D Shape benchmark ModelNet40, and measure how it is impacted by voxel resolution of input shape. We demonstrate comparable classification and retrieval performance to state-of-the-art models, but with much less computational costs in training and inference phases. We also notice that benefits of higher input resolution can be limited by an ability of a neural network to generalize high level features.

1 Introduction

For computer vision systems a precise and robust operations in real environments is only possible by harnessing information from 3D data. To achieve this we need to overcome some challenges of this kind of problems.

Data, received from such devices as $2.5D$ scanners, often are given in the form of noisy meshes or point clouds, which is not the best fit for new kinds of models such as Convolutional Neural Networks (CNN's) [13].

In the current state-of-the-art systems Convolutional Neural Networks are widely used, so their effectiveness in processing 2D images is also suggesting that CNNs should be efficient to process 3D objects if presented in the form of several rendered views of the object. For example on one ModelNet40 [22] benchmark three recent papers, based on this idea, showed incremental improvements in recognition performance [18,9,7]. However, it can be argued that high performance is predicated by the usage of CNNs pre-trained on ImageNet [5].

Voxel representation of 3D shapes (i.e. a shape is represented as a three-dimensional grid, where occupied cells are binary values) are compatible with ConvNets input layers but create a number of difficulties. Adding a third spatial dimension in the input grid correspondingly increases computational costs. Number of cells scales as a power of three w.r.t. the resolution of the voxel grid. Low resolution grids make it difficult to differentiate between similar shapes, and lose some of the fine details available in 2D renderings of equivalent resolution.

Some 3D Dense Convolutional Networks have been evaluated on the ModelNet40 benchmark [15,17], but they still not perform as well as their multi-rendering 2D counterparts.

At the same time using Modified Spatially Sparse Neural Networks algorithms [6] to process data we are able to have reasonable training and inference time even with input resolution up to 100^3 voxels.

In this work, we present Sparse 3D Deep Convolutional Neural Networks and explore their ability to perform large-scale shape retrieval on the popular benchmark ModelNet40 [22] depending on an input resolution and a network architecture.

Sparse 3D CNNs are able to generate relevant features for retrieval analogously to 2D extractors. To have a system that uses many 2D rendered projections for inference is computationally very costly, especially for the task of Large Scale 3D Shape Retrieval. In this paper we present some preliminary results of our attempt to find out if the resolution of an object Voxelization impacts on descriptive feature extraction as measured by the retrieval performance on a sufficiently big dataset. Also we demonstrate ability of Sparse 3D CNNs to perform metric learning in the triplet loss setup. Lastly we train our model to perform classification on the ModelNet40 benchmark.

In Section 2 we formulate the problem in more detail and discuss latest relevant methods. In Section 3 we describe our approach for neural nutworks that helps us solve the problem posed in Section 2. In Section 4 we document conditions of computational experiments we performed. In Section 5 we discuss results and make conclussions about our approach to problem.

2 3D Large-Scale Retrieval

2.1 Large-Scale 3D Shape datasets

As can be seen the great improvements in recent years for the problem of 2D large-scale image recognition, are not just the result of wide-spread adoption of Deep Learning techniques, but also it is due to the availability of large datasets that capture sufficient variety of features at different scales to be representative of some domain. However, only recently in the 3D recognition and retrieval such datasets started being published.

The recent competition [16] evaluated several models utilising Neural Networks for 3D retrieval on ShapeNet Core55, which is a subset of ShapeNet [4] — a dataset with more than 50 thousand models in 55 common object categories. The approach for creating descriptors from multiple projections of a 3D shape with a transfer learning from ImageNet showed the best performance [18]. No full 3D algorithms that process voxels directy have been described up to now.

The other great example is the ModelNet [4] dataset. ModelNet40 is a subset of this dataset, and it is going to be our main benchmark for the retrieval task.

2.2 Shape descriptors

To make inferences about 3D objects for purposes of computer vision or computer graphics, researchers developed a big amount of shape descriptors[10,11,3,12].

Shape descriptors usually fit into two categories: one where shape descriptors are computed using 3D representations of objects, e.g. voxel discretizations, meshes, point clouds, or implicit surfaces, and the second one that describes a shape of a 3D object by a collection of 2D projections, often from multiple viewpoints.

Before large-scale 3D shape datasets such as ModelNet [22] and 3dShapeNet model which learns shape descriptors from voxel representation of a mesh object through 3D convolutional nets, 3D shape descriptors were mostly special functions capturing specific geometric properties of the shape surface or volume, for example: spherical functions computed on volumetric grids [10], generalization of SIFT and SURF feature descriptors for voxel grids [11], or for non-rigid bodies and deformable shapes heat kernel signatures on meshes [3,12]. Developing classifiers and other supervised machine learning algorithms on top of such 3D shape descriptors poses a number of challenges. The success of CNNs image descriptors allows us to hope that descriptors based on 3D convolutional nets can be also beneficial compared to classic descriptors.

2.3 Triplet learning

Recent work in [8] shows that learning representations with triplets of examples gives much better results than learning with pairs using the same network. Inspired by this, we focus on learning feature descriptors based on triplets of patches.

Learning with triplets involves training from samples of the form (a, p, n), where

- a is an anchor object,
- p denotes a positive object, which is a sample we want to be closer to a and usually being a different sample of the same class as p, and
- n is a negative sample belonging to a different class than a and p.

Optimizing parameters of the network brings a and p close in the feature space, and pushes apart a and n.

Finally, let us introduce this triplet loss, also known as the ranking loss. It was first proposed for learning embedding using CNNs in [21] and can be defined as follows:

- Let us define $\delta_+ = \text{cosine}(f(a), f(p))$ and $\delta_- = \text{cosine}(f(a), f(n))$, i.e. this is a cosine distance between some feature representations $f(\cdot)$ for different objects,
- Then for a particular triplet we calculate the triplet loss using the formula

  $$\lambda(\delta_+, \delta_-) = \max(0, \mu + \delta_+ - \delta_-)\,,$$

  where μ is a margin parameter. The correct order should be $\delta_- > \delta_+ + \mu$,
- If order of objects, provided by their corresponding descriptors are incorrect w.r.t. the triplet loss, then the network adjusts its weights through back-propagation signal to reduce the error.

3 Sparse Neural Networks

Using sparsity to make a neural network computations more efficient is pioneered by Benjamin Graham [6], who developed a low-level C++/CUDA library SparseConvNet[3] that implements strided convolutions and max-pooling operations on a D-dimensional sparse tensors using GPU. Thanks to this inherited sparsity we are able to process data in reasonable training and inference time even with input resolution up to hundreds of voxels. More precisely an information about voxels in a given layer is not stored in a 3-dimensional array, but in a sparse vector with active cells as elements.

Transformation of data between layers (e.g. convolutions, pooling, nonlinear activation functions), are performed on those sparse vectors. Data in areas with inactive voxels, which are most of them, does not depend on a voxel relative position, therefore it can be replaced by vectors of a smaller size without explicit spatial dimensions.

It's well known that, operating with a sparse data structures is less efficient than working with dense data. Another useful property is that we don't need to store much less data for each object. We have computed sparsity for all classes of ModelNet40 train dataset at voxel resolution equal to 40, and it's only 5.5%.

Paper [22] describes using 3D convolutions for their deep model. Voxel labeled as active when it's intersects with a mesh object, and inactive otherwise. This binary representation of 3D shape given as input to a 3D CNN, which has a structure similar to a 2D one. The main problem of this approach is ineffectiveness with which data is represented and processed. Mentioned model uses 30^3 cells, which is approximately the number of pixels in 2D applications of CNN. If we take into account linear dimensions it's obviously not a lot, as can be seen from Figure 1. That resolution was primarily chosen because of computational resource limitation. Besides that, — convolution is very computationally expensive operation, complexity of which rises very fast with input scale. Computational complexity of 3D convolution for image with dimensions of $N \times M \times K$ with filters sizes of $n \times m \times k$ is equal to $\mathcal{O}(NMKnmk)$. If we use Fast Fourier Transform (FFT), complexity can be reduced to $\mathcal{O}((N+n)(M+m)(K+k)\log((N+n)(M+m)(K+k)))$ in exchange for more memory cost [14]. But even in that case, complexity of convolutions makes it impossible to work with objects in big voxel resolutions.

3.1 PySparseConvNet

The SparseConvNet Library is written in C++ programming language, and utilizes a lot of CUDA capabilities for speed and efficiency. But it is very limited when it comes to

- extending functionality — class structure and CUDA kernels are very complex, and require re-compilation on every modification.

[3] `https://github.com/btgraham/SparseConvNet`

- changing loss functions — the only learning configuration was SoftMax with log-likelihood loss function.
- fine grained access to layer activations — there were no way to extract activations and therefore features from hidden layers.
- interactivity for exploration of models — every experiment was a compiled binary with now way to perform operations step by step, to explore properties of models.

Because of all these problems we developed PySparseConvNet[4]. On implementation level it's a python compiled module that can be used by Python interpreter, and harness all of it's powerful features. Most of modern Deep Learning tools, such as [19,1,20] using Python as a way to perform interactive computing.

Interface of PySparseConvNet is much simpler, and consist's of 4 classes:

- **SparseNetwork** — Network object class, it has all the methods to changing it's structure, manipulate weights and activations.
- **SparseDataset** — Container class for sparse samples and their labels.
- **SparseBatch** — Gives access to data in dataset when processing separate mini-batches.
- **Off3DPicture** — Wrapper class for 3D models in OFF (Object File Format), used to voxelize samples to be processed by SparseNetwork.

layer #	layer type	size	stride	channels	spatial size	sparsity (%)[5]
0	Data input	-	-	1	126	0.18
1	Sparse Convolution	2	1	8	125	-
2	Leaky ReLU ($\alpha = 0.33$)	-	-	32	125	0.35
3	Sparse MaxPool	3	2	32	62	0.69
4	Sparse Convolution	2	1	256	61	-
5	Leaky ReLU ($\alpha = 0.33$)	-	-	64	61	1.07
6	Sparse MaxPool	3	2	64	30	1.93
7	Sparse Convolution	2	1	512	29	-
8	Leaky ReLU ($\alpha = 0.33$)	-	-	96	29	3.26
9	Sparse MaxPool	3	2	96	14	7.32
10	Sparse Convolution	2	1	768	13	-
11	Leaky ReLU ($\alpha = 0.33$)	-	-	128	13	15.14
12	Sparse MaxPool	3	2	128	6	46.30
13	Sparse Convolution	2	1	1024	5	-
14	Leaky ReLU ($\alpha = 0.33$)	-	-	160	5	97.54
15	Sparse MaxPool	3	2	160	2	100.00
16	Sparse Convolution	2	1	1280	1	-
17	Leaky ReLU ($\alpha = 0.33$)	-	-	192	1	100.00

Table 1: S3DCNN Network architecture.

[4] `https://github.com/gangiman/PySparseConvNet`

[5] Last column "sparsity" is computed for render size = 40 and averaged for all samples

4 Experiments

4.1 Implementation details

To demonstrate the impact that the triplet based training has on the performance of CNN descriptors we use a deep network architecture shown in a Table 1. This network was implemented in PySparseConvNet, which is our modification of the SparseConvNet library [6]. Besides new loss functions PySparseConvNet can be accessed from Python for a more interactive usage.

When forming a triplet for training we choose uniformly randomly a positive pair of objects from one class and select a negative sample uniformly randomly from one of other classes.

For the optimization we use the SGD [2], and the training is done

- in batches of size from 45 to 90 depending on a GPU video memory,
- with a learning rate of 0.002,
- and a momentum equal to 0.99.

Training can take up to a week on a server with advanced GPU, such as NVIDIA Titan X or GTX980ti.

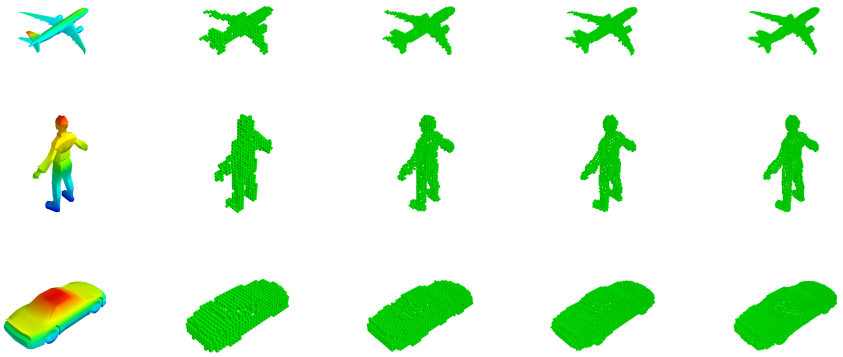

Fig. 1: Examples of some objects voxelizations at different resolutions 30, 50, 70, 100 (from left to right), left-most objects are depicted using original meshes

We train Sparse 3D Convolutional Neural Network (S3DCNN) on the 3D shape classification dataset by splitting it into training and validation subsets, adding augmentation of data to achieve rotational and translational invariance. After training a model on a dataset of pairs, we use it to embed voxel representations of 3D meshes into 192-dimensional space. The retrieval consist of ranking search objects by a cosine distance of vectors from a query vector.

The most popular metrics for evaluating retrieval performance are

- Precision-Recall Curve shows a trade-off between these two measures and how quickly the precision drops with the recall increase,
- Mean average precision (mAP). Given a query, its average precision is the average of all precision values computed on all relevant objects in the retrieved list. Given several queries, the mean average precision (mAP) is the mean of average precisions for these queries.

We evaluated mAP for different voxel rendering sizes of 3D shapes both at train and test times, see also Figure 1.

To check if our model is comparable with other architectures, we consider a classification task. So, we trained our model for the classification task using the ModelNet40 train subset with

- SoftMax last layer for 200 epochs,
- with exponentially discounting learning rate,
- and performed retrieval evaluation on the test subset,
- taking 20 images from every class, and ranking them w.r.t their $L2$-norm by activations taken from the 17-th layer.

Results of these experiments are provided in Table 2. We can see that in case of classification task setup our model is comparable in terms of the classification accuracy, but mAP values are worse. But in case of metric learning performace of S3DCNN on mAP metric is much better. Superior performance of retrieval task with MVCNN is not a surprising result, since MVCNN uses neural nets, pre-trained on ImageNet. On the other hand our model only requires 3D Shape dataset to learn.

In Figure 2 we provide the dependence on mAP on the input spatial resolution. We can see that the retrieval performance improves with increase in the input spatial resolution up to around $45 - 50$, after that it drops slightly and plateau's. It can be attributed to the insufficient amount of layers for the same scale of features, that can be separated in higher layers. Light blue color shows range of mAP on validation for top 30 trained architectures.

We would like to note that in Figure 2 mAP values provided for different validation epochs and variablity of best model can be explained by difference in total learning time.

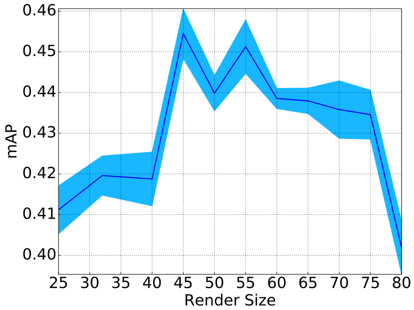

Fig. 2: Dependence of the retrieval performance on the input spatial resolution

5 Results

We found that the retrieval performance improves with increase in the input spatial resolution. However, such an effect is difficult to check experimentally and to use in practice, as e.g. for usual 3D dense CNNs the computational time is prohibitively large. In

Table 2: Evaluation on Modelnet40

method	Classification	Retrieval AUC	Retrieval mAP
3DShapeNet	77.32%	**49.94**%	49.23%
MVCNN	90.10%	—	**80.20**%
S3DCNN (proposed)	**90.3**%	36.05%	33.67%
S3DCNN + triplet (proposed)	—	48.81%	46.71%

our case thanks to the sparsity we can process data in reasonable time even with input resolution up to 100^3 voxels, therefore we can benefit from the increase of the input spatial resolution when performing retrieval. On Figure 3 you can see that our method comparable to [22] in low recall, and better at higher recall values, that indicates better scalability of our method. We would like to perform more thorough optimization of learning hyper-parameters to achieve better performance. We provide training code for all experiments in our repository[6].

Acknowledgments We are very grateful to Dmitry Yarotsky for his contribution to this research project. Big Thanks to Benjamin Graham for some useful comments and ideas. Thanks to Rasim Akhunzyanov for his help in debugging the PySparseConvNet code.

The research was partially supported by the Russian Science Foundation grant (project 14-50-00150).

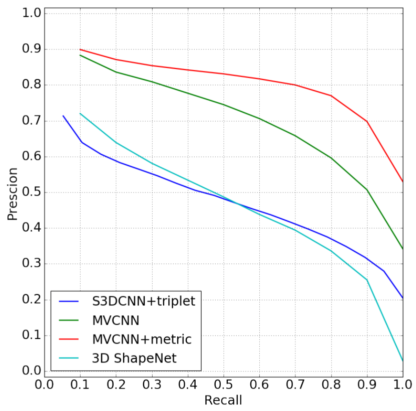

Fig. 3: Precision-Recall curve for our method

[6] https://github.com/gangiman/PySparseConvNet

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, and I. G. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
2. L. Bottou. Stochastic gradient tricks. *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445, 2012.
3. A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1, 2011.
4. A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
5. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
6. B. Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.
7. V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.
8. E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.
9. E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. *arXiv preprint arXiv:1605.08359*, 2016.
10. M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.
11. J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *European Conference on Computer Vision*, pages 589–602. Springer, 2010.
12. I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein. Intrinsic shape context descriptors for deformable shapes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 159–166. IEEE, 2012.
13. Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
14. M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.
15. D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.
16. M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai, et al. Shrec'16 track large-scale 3d shape retrieval from shapenet core55.
17. N. Sedaghat, M. Zolfaghari, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016.
18. H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.
19. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

20. S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.
21. J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.
22. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.