3D Object Recognition with Deep Networks

Adrian Schneuwly
adrischn@ethz.ch

Johannes Oswald
voswaldj@eth.ch

Tobias Grundmann
tobiagru@ethz.ch

Abstract

The increasing amount of LiDAR and RGBD cameras in e.g. mobile devices and robotic systems provide us with new information which can aid in various task including object recognition. In this paper we will discribe 3D Convolutional Deep Neural Networks and how they are trained to recognize voxelized point cloud data from common objects.

1. Introduction

One of the crucial tasks of computer systems based on visual information e.g. self-driving cars, autonomous robots, virtual enviroments (Figure 1) is to get a semantic understanding of the enviroment. Depth data has already proved its usability in obstacle avoidance or mapping but we want to investigate its potential to improve object recognition.

Since Deep Learning dramatically improved state of the art object recognition [4], we will use Neural Networks to recognize objects from given point clouds. Especially the idea behind convolution in 2D Object recognition will be translated to our 3D data, resulting in a fast and accurate detector.

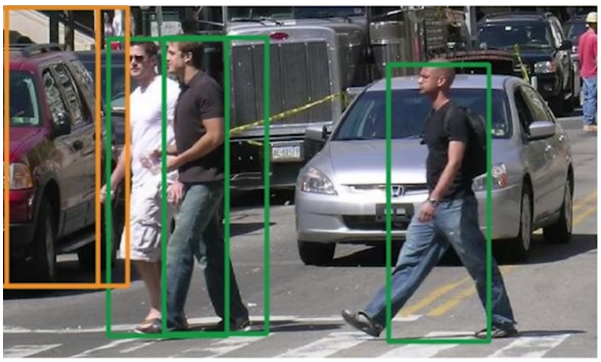

Figure 1. Example of object recognitions. Source: [3]

2. Related Work

On the one hand, There is a large body of work of object recognition using 3D point clouds but not making use of Neural Networks. Mostly methods combine hand-crafted features or descriptors with a machine learning classifier ([10], [11], [12]). Similiar methods are seen with semantic segmentation, with structured output classifiers instead of single output classifiers ([14], [15], [16]).

On the other hand 2.5D CNNs were used for object recognition but fail to make full use of geometric information in the data. Their approaches simply treat the depth channel as an additional channel, along with the RGB channels. ([17], [18], [19], [20])

Additionally, 3D CNNs already proved there usability in video analysis ([23], [24]). In this case, time acts as the third dimension but algorithmically, these architectures work the same as ours, but the nature of the data is very different.

Das so machen? ist quasi kopiert von VoxNet. Finde ich ok, mir egal

3. Convolutional Neural Networks

At the core of a neural network is a logistic classifier. For given data, we train weights and bias of a linear equation which will then be transformed into probabilites by using a softmax function.

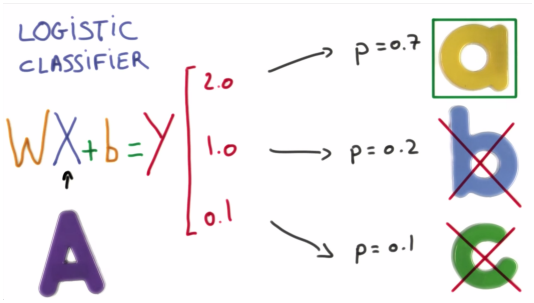

Figure 2. Weight and bias (W,b) get trained to classify input A. Source: [3]

Linear models are limited and therefore we deepen the network in our case through convolutions and max pooling. For convolution we define a patch which then walks through our 3D Data and runs a small neural network on this patch with given output size / depth. When leaving out /jumping voxels (striding), the spatial dimension is squeezed while extracting local features of the given object. Through re-

peating this procedure we arrive at an output depth which coresponds to a similiar semantic complexity of our representation.

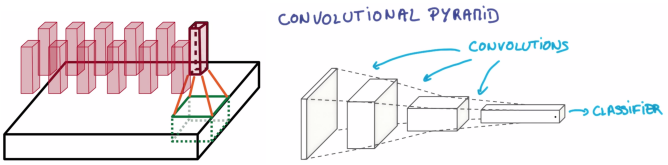

Figure 3. Left: Patch with stride 2 Right: Multiple Covolutions. Source: [3]

The second technique to deepen our network is pooling. Instead of striding to sqeeze our dimension, we can extract e.g. the maximum of a patch and therefore reduce spatial dimension. See detailed discribition of model used in our approach in Figure 5.

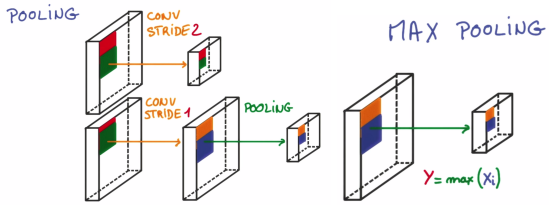

Figure 4. Left: Stride 2 vs. Stride 1 + Pooling Right: Max-Pooling. Source: [3]

In order to train our model, we have to test how well it is doing during and after training. Testing the model on the data we trained it with does not work since the model remembers input data so we split the data into test- and training set which gives us accurate results how well the classifier is doing on unknown data. This common phenomenon, that a complex model fits a given dataset but fails to generalizes on new data, is called *overfitting*.

4. Data: Princeton ModelNet

The data on which we train our Deep Network was build by [6]. After a list of the most common object categories in the world was compiled from Princtons SUN database [2], 3D CAD models were collected through online search engines. To verify the object assigment to each category, human workers on Amazon Mechanical Turk were hired to validate manually. We trained our neural network on one smaller and one larger dataset containing 10 respectivally 40 categories which each ?XXXX? objects.

5. Reimplementing: VoxNet [5]

In the following, parts contributed by Author 1/2/3 will be labeled as A1/A2/A3.

Overall Goal is to classify an object from a given point cloud. After voxilizing to format 32x32x32, we train our CNN on these occupancy grids, Figure 5.

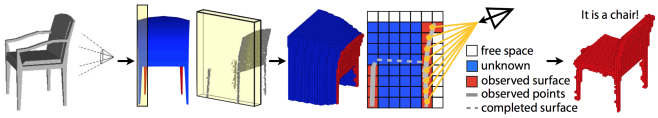

Figure 5.
Object - Depth & Point Cloud - Occupancy Grid - Classification. Source: [6]

Princetons ModelNet delivers every object with views from 12 different angles in .mat files; we train on all rotations to enhance recognition from different viwes. The data is already voxilized to 32x32x32 occupancy grids. This data was transformed by A3 into a .hdf5 format and split into test and training set. Hdf5 compresses the set size dramatically and enables simple access. Two methods were implemented by A1/A3 to deal with loading the data from .hdf5 files. One opens the .hdf5 and loads pieces which are passed during iterations of the generator resulting in a slower but RAM saving loader. The other, used for the final training, converts the hdf5 entirely to a numpy array providing us with a RAM expensive but much faster access to the data.

After preparing the data and its access, the VoxNet convolutional neural network model (900.000 Parameters) is created, Figure 5. For this, A1/A2 used the *Python* deep learning library *Keras* which runs on top of *Theano*. *Keras* supports 3D convolutional and max-pooling layers.

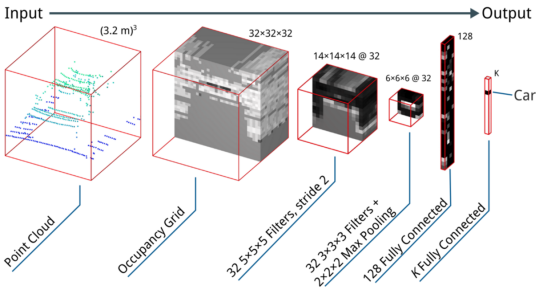

Figure 6. VoxNet Convolution Model. Source: [1]

A training enviroment was set up by A1/A2/A3.

6. Experiments

The training process takes around 9 to 20 hours on a NVIDIA GTX 980TI (6GB) GPU depending on the choice of the dataset (ModelNet10/40).

7. Results

The papers implementation achieved verg good results for the small data set but failed to achieve the excelent score of the original VoxNet implementation on the larger ModelNet40. This is due ... TODO? WARUM?

Algorithm	ModelNet10 Classification Accuracy	ModelNet40 Classification Accuracy
VoxNet [5]	83%	92%
3DShapeNets [6]	77 %	83.5%
ETH VoxNet	**81.8%**	**82.3 %**

8. Conclusion

The team successfully reimplemented a very clean and working VoxNet with comparable results.

References

[1] D. maturana website. `http://www.dimatura.net/pages/3dcnn.html`.

[2] Sun database. `http://sun.cs.princeton.edu/`.

[3] Udacity deep learning. `https://www.udacity.com/course/deep-learning--ud730`.

[4] J. S. A. S. Razavian, H. Azizpour and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CoRR, vol. abs/1403.6382*, 2014.

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

[6] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *CVPR*, 2015.