Understanding 3D Space-Semantics

Iro Armeni

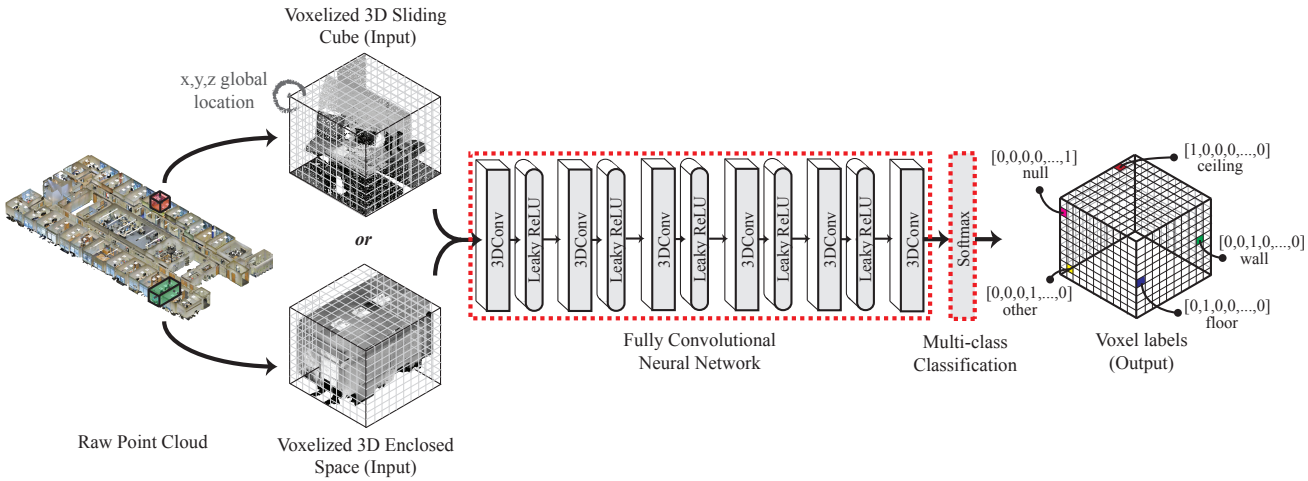

Figure 1: **Frameworks for 3D Parsing of Large-Scale Indoor Point Clouds into their Space-Semantics.** Exploring two different network architectures: (1) A fully 3D CNN receives as an input a 3D voxelized sliding cube with binary occupancy and performs a per voxel multi-class classification into 5 semantic labels. (2) A fully 3D CNN receives as an input a voxelized enclosed space with binary occupancy and performs a per voxel multi-class classification into 10 semantic labels. The result in both cases is a class prediction per output voxel.

Abstract

Point clouds are comprehensive and detailed 3D representations for indoor spaces, however they contain no high-level information of the depicted area. In contrast to previous methods that have mostly focused on more traditional pipelines, I propose a Fully 3D Convolutional Neural Network for the semantic parsing of such data. In this paper I explore different network architectures and perform per voxel multi-class classification of the input into its semantics. Two main inputs are explored: (a) a 3D sliding cube on the large-scale point cloud and (b) a consistently aligned and normalized enclosed space of the point cloud (i.e. rooms, hallways, etc.). Both inputs are voxelized with values equal to their binary occupancy. The network outputs a class prediction per voxel. I provide experiments and results on the different architectures and used the Stanford Large-Scale 3D Indoor Dataset for their evaluation.

1. Introduction

We spend 90% of our time indoors [3] and for systems to operate in this environment (*e.g.* assist us in daily activities) they need to have an understanding of it. 3D depth sensors are quickly becoming standard practice and have enabled the representation of our surroundings as whole scenes in an ever increasing number of point clouds. Although such data is becoming more and more available, it is not directly useful since it does not contain high-level information of the depicted elements, as e.g. space-semantics. Such an understanding would be beneficial to many applications related to augmented reality, robotics, graphics, the construction industry, etc.

Previous work has focused on extracting this information from point cloud data by following semantic parsing approaches (*e.g.* segmentation or object detection). However the great majority of them resorts to more traditional pipelines [26], where hand-engineered features are extracted and fed into an off-the-shelf classifier such as Support Vector Machines (SVMs). The past few years these methods are gradually being abandoned in favor of Deep Learning ones that produce superior results by learning features and classifiers in a joint manner. Convolutional Neural Networks (CNNs) in specific have made great progress and are widely used especially in the case of 2D data. Nevertheless, the area of CNNs on 3D data, especially for the task of parsing, is significantly less explored to its 2D counter-

parts [18].

Most work in the context of 2.5D and 3D using ConvNets is targeting other applications like depth from RGB [7], camera registration [13], and human action recognition [11]. This has limited the amount of produced knowledge and available implementations, pretrained models and training data for 3D related tasks. Alternative approaches to the problem of detecting 3D space semantics in large-scale indoor point clouds could be formed by posing the problem as 2D or 2.5D. Although these could benefit from pretrained models, existing architecture and other 2D or 2.5D datasets for training (*e.g*. [27] and [22] respectively), they would not take advantage of the rich spatial information provided in 3D point clouds, which can help disambiguate problematic cases. It has been shown that 3D parsing methods can perform better than their 2.5D counterparts [5].

I propose instead a framework for the task of parsing 3D point clouds of large scale indoor areas into their space-semantics using an end-to end 3D CNN approach. In a higher level, the network receives as an input a voxelized 3D portion of a large-scale point cloud [1] and through a series of fully $3D$ convolutional layers it performs multi-class classification on the voxel level, and outputs the predicted class for each voxel. The network classifies each input voxel into 10 semantic labels[2] related to structural and building elements, clutter and empty space.v

2. Related Work

Traditional Approaches: Semantic RGB-D and 3D segmentation have been the topic of a large number of papers and have lead to a considerable leap in this area during the past few years. For instance [29, 24, 22] propose a RGB-D segmentation method using a set of heuristics for leveraging 3D geometric priors. [21] developed a search-classify based method for segmentation and modeling of indoor spaces. These are different from the proposed method as they mostly address the problem in a small scale. A few methods attempted using multiple depth views [28, 9], but they remain limited to small scale. Unlike approaches such as [26], [15], my method learns to extract features and classify voxels from the raw volumetric data. Vote3D [31] proposes an effective voting scheme to the sliding window approach on 3D data to address their sparse nature.

2.5**D Convolutional Neural Networks:** A subsequent extension to RGB-D data followed the success of 2D CNNs ([17], [30], [4], [10]). However, most work handles the depth data as an additional channel and hence it does not make full use of the geometric information inherent in the 3D data. [8] proposes an encoding that makes better use of the 3D information in the depth, but remains 2D-centric. The presented work differs from these in that I employ a fully volumetric representation, resulting in a richer and more discriminative representation of the environment.

3**D Convolutional Neural Networks:** 3D convolutions have been successfully used in video analysis ([11], [12]) where time acts as the third dimension. Although on an algorithmic level such work is similar to the proposed one, the data is of very different nature. In the RGB-D domain, [16] uses an unsupervised volumetric feature learning approach as part of a pipeline to detect indoor objects. [32] proposes a generative 3D convolutional model of shape and apply it to RGB-D object recognition, among other tasks. VoxNet [20] presents a 3D CNN architecture that can be applied to create fast object class detectors for 3D point cloud and RGB-D data. This work has similarities, however among the differences: it uses a different input representation, it is not performing voxel-to-voxel classification and since the task is detection it uses fully connected layers.

3. Method

The proposed method receives as an input a voxelized 3D portion of the point cloud and through a series of 3D convolutions results to a class label prediction for each voxel. I gradually explored 3 different approaches:

- **3D Sliding Window:** In this network, the input is a voxelized 3D cube of constant size with binary occupancy that is sled over the large-scale point cloud. When fed to the network, it passes through a series of 3D Fully Convolutional layers which result to a per-voxel multi-class classification (see Figure 1-Left).

- **Adding Context:** The previous approach does not provide any context about the content of the sliding cube in relation to the rest of the point cloud. However, context can strongly influence inference. To this end, I provide the global position of the sliding cube in the point cloud as a second input to the network, following a similar approach to [6] (see Figure 1-Right).

- **Enclosed Spaces:** The use of a sliding cube with constant size cannot account for the different sizes that elements in the point cloud appear with. Although for elements that belong to the category of *things* (*e.g*. chairs or tables) one can learn a dictionary of shapes, for elements that can be categorized as *stuff* (*e.g*. walls or ceiling) it is harder to identify repetitive shape and size patterns. To address this issue I explored an approach similar to [5] to take advantage of the repetitive layout configuration that indoor enclosed spaces present (*e.g*. elements are placed in a consistent way inside a room with respect to the entrance location). The semantics in such spaces remain intact (*e.g*. the wall, ceiling and

[1] The scale of the point cloud can range from a whole building to a floor, or any large portion of the former.

[2] Due to memory restrictions some of the presented experiments use either 5 labels. For more details see 4.3.

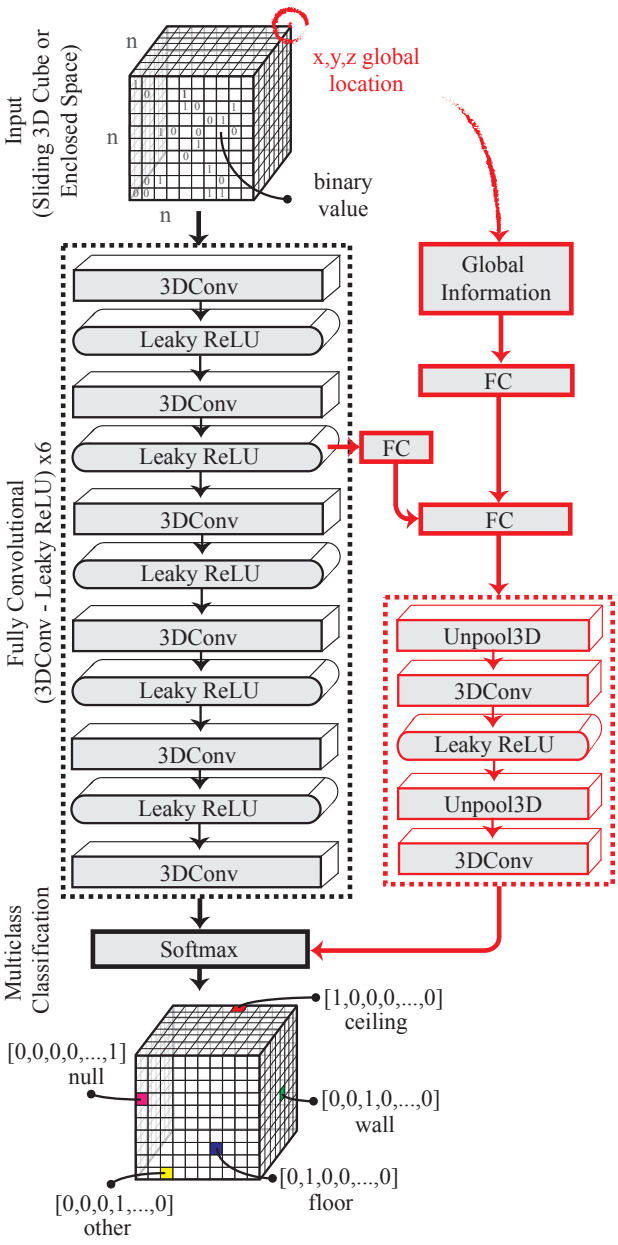

Figure 1. 3**D Convolutional Neural Network Architecture.** *Left:* 3D Sliding Window or Enclosed Space.*Right:* Adding Contextual Information.

other elements appear in the data as a whole). These spaces are voxelized, consistently aligned and normalized in a unit cube before they are fed into a 3D Fully Convolutional Neural network. The task is again that of per voxel classification (see Figure 1-Right).

In the remainder of this section I will explain the main network architecture and offer details about each of the above-mentioned approaches.

3.1. Input

The input to the method is a voxelized cube that represents the underlying spatial data with binary occupancy.

3D Sliding Window: Here the input is a 3D sliding cube of size $10x10x10$ voxels. The stride of the cube on the point cloud in all 3 dimensions is 10 voxels, which means that there is no overlap between spatially consecutive cubes [3] The size of the input is heuristically defined and takes into consideration the voxel resolution and space-structure. In specific, I targeted to capture the representation of walls and room borders in point clouds as an empty space in between two surfaces. To reflect that in the voxelized space we chose a voxel resolution of $5x5x5$cm, since the standard minimum wall width is 7-10cm. As a result, a sliding cube of $10x10x10$ voxels corresponds to $50x50x50$cm in space, which can encompass both the minimum wall width and a gap between rooms larger than the standard wall size (either due to noise or occlusion of the wall surfaces by *e.g.* a bookcase in highly cluttered scenes).

Adding Context: In this approach a second input to the voxelized sliding cube is fed to the network, which is the global location of the cube with respect to the whole point cloud. This is represented by its x, y, z coordinates from a defined starting point (one of the point cloud's corners) and forms a vector of size 3.

Enclosed Spaces: As mentioned above the input here is an enclosed space. One can segment the point cloud into such spaces with a variety of different approaches ([5], [23]). Once these spaces are identified, they are projected to a canonical reference coordinate system. In this reference system all spaces are systematically aligned with respect to their entrance location and subsequently normalized in a unit cube. Before they are fed to the network, they are voxelized with binary occupancy values. Due to memory restrictions, the resolution of the voxelization was selected as $0.2x0.2x0.2$, thus forming an input of size $50x50x50$ voxels.

3.2. Fully Convolution 3D Neural Network

The input is then fed into a 3D Fully Convolutional Neural Network. The choice of not including any Fully Connected layer is based on the task in hand; since we are performing a voxel to voxel operation, retaining the spatial information throughout the network is considered essential. The network comprises of the following repetitive unit: a 3D Convolutional Layer (3D Conv) followed by a Leaky Rectified Linear Unit (ReLU) [19] (apart from the last layer). The choice of Leaky ReLU over e.g. ReLU is

[3]The size of the stride as well as other network details decisions have been largely driven by memory limitations. An overlapping stride would allow to infer the class label of each voxel not only based on its neighbors in one cube location, but also by taking into account a larger area around it.

Table 1. Details of 3D Fully Convolutional Neural Network

Approach	3D Sliding Cube	Enclosed Space
	Input	
Size:	$Nx10x10x10$	$Nx50x50x50$
Number of Channels:	1	1
Stride:	$10x10x10$	
-		
	3D Conv1	
Number of Filters:	32	
Filter Size:	$5x5x5$	
Stride:	$1x1x1$	
Padding:	$2x2x2$	
Output Size:	$Nx32x10x10x10$	$Nx32x50x50x50$
	3D Conv2 and 3D Conv3	
Number of Filters:	64	
Filter Size:	$5x5x5$	
Stride:	$1x1x1$	
Padding:	$2x2x2$	
Output Size:	$Nx64x10x10x10$	$Nx64x50x50x50$
	3D Conv4 and 3D Conv5	
Number of Filters:	128	
Filter Size:	$5x5x5$	
Stride:	$1x1x1$	
Padding:	$2x2x2$	
Output Size:	$Nx128x10x10x10$	$Nx128x50x50x50$
	3D Conv6	
Number of Filters:	5	10
Filter Size:	$5x5x5$	
Stride:	$1x1x1$	
Padding:	$2x2x2$	
Output Size:	$Nx5x10x10x10$	$Nx10x50x50x50$
	Output	
Size:	$Nx5x10x10x10$	$Nx10x50x50x50$

made to avoid saturated neurons. Mathematically, we have $y = x_i$ if $x_i \geq 0, else$ y =x_i/a_i, where a_i is a fixed parameter in the range $(1, \infty)$. I followed the original paper's configuration and set a_i to 100. For this experiment I used 6 layers [4], which details are tabulated in Table 1 1.

Adding Context: In this approach we use a double input to the network and follow a similar architecture to the one proposed in [6]. The voxelized cube passes first through 2 3D Convolutional layers to encode spatial information and then gets concatenated with the global position. The concatenated vector passes first through a Fully Connected layer and subsequently through layers of deconvolution to acquire again its spatial configuration. For more details see Figure 1.

3.3. Multi-class Voxel Classification

At the end of the final 3D Convolutional Layer the network performs multi-class Softmax classification and will predict the scores of each class label per voxel: $f_j(z) = e^{z_j} / \sum_k e^{z_k}$, where z is each voxel, j is the class evaluated and k represents all classes.

[4]The number of layers and filters in the network was a direct result of the memory limitations.

4. Experiments

4.1. Dataset

For the evaluation I used the Stanford Large-Scale 3D Indoor Dataset [5] which comprises of six large indoor parts in three buildings of mainly educational and office use (see Figure 2). The entire point clouds are automatically generated without any manual intervention as the output of the Matterport camera ([1]). Each area covers approximately 965, 1100, 450, 870, 1700 and 935 square meters (total of 6020 square meters). Conference rooms, personal offices, auditoriums, restrooms, open spaces, lobbies, stairways and hallways are commonly found. The areas show diverse properties in architectural style and appearance. The dataset has been annotated for 12 semantic elements which pertain in the categories of structural building elements (ceiling, floor, wall, beam, column, window and door) and commonly found furniture (table, chair, sofa, bookcase and board). A clutter class exists as well for all other elements. The dataset was slpit into training, validation and testing as follows: 4 areas for training, one for validation and one for testing. In this way I ensure that the network sees areas from different buildings during training and testing. The same data split was used in all approaches.

4.1.1 Preprocessing

3D Sliding Cube: The raw colored point clouds were voxelized in a grid of $5x5x5$cm (for more details on the choice of grid resolution see Section 3.1). After voxelization I assigned binary occupancy to all voxels as 0 if the voxel is empty, 1 if occupied. The ground truth labels were generated by finding the mode of all point labels per voxel. Due to memory restrictions, the number of classes was limited to 5: walls, floor, ceiling, other and empty space.

Enclosed Spaces: Similarly, the aligned point clouds that correspond to each enclosed space were normalized in a $0.2x0.2x0.2$ grid. The generated voxels was the populated with binary occupancy values. The ground truth labels were generated as in the previous case. Due to memory restrictions, the number of classes was limited to 10: walls, floor, ceiling, door, beam, column, chair, table, other, and empty space.

4.2. Implementation

I performed three different experiments, one for each input approach. The implementation details are:

- I implemented the framework in Python3 using the deep learning Python library Theano [2].
- All data preprocessing steps were implemented in Python3 as well.

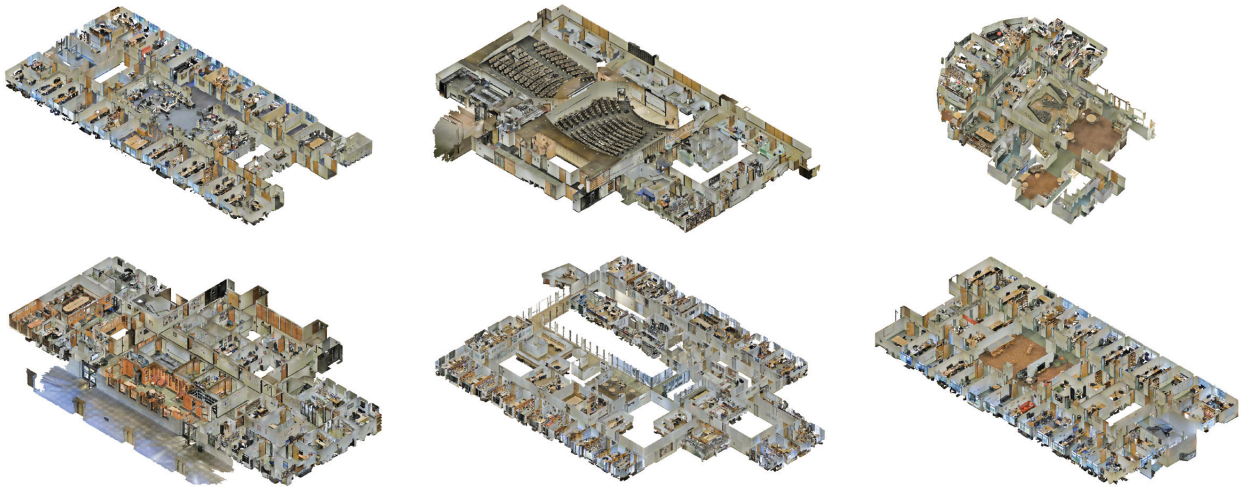

Figure 2. **Stanford Large-Scale 3D Indoor Dataset [5]:** I split the dataset into training (4 areas), validation (1 area) and testing sets (1 area). The raw point clouds are shown in the first row and the voxelized ground truth ones in the second row.

- **3D Sliding Cube:** After generating the input, I noticed that the amount of sliding cubes that contained only empty voxels was greatly larger than the sliding cubes that contained at least one voxel of the rest of the classes. To counterpoise the skewness of the distribution I removed part of the empty sliding cubes.

- I shuffled the data before training since without it the learning process was getting compromised. The network was receiving sequentially inputs of similar classes in the first case due to the sliding nature of the input and the semantic consistency of the configuration of spaces and in the second due to spaces with similar functions.

- I used the Adam [14] adaptive learning rate method, with parameters: 0.9, 0.99, and $1e-08$.

- The size of the batch per iteration was limited to 500 sliding cubes for the first and second approaches (sliding cube) and 4 for the third (enclosed space) due to memory restrictions.

- I used as metric the mean accuracy of prediction per voxel.

4.3. Results

3D Sliding Cube: The initial idea towards this problem was to use a 3D sliding window approach. The main motivation behind it was the fact that the input size to the network and the size of the voxelization grid could remain constant no matter the size of the point cloud (buildings have different sizes). Previous experience with this framework in traditional pipelines has been shown successful. However, although substantial effort was put to tune the hyper parameters, the network did not learn. I identify four main factors as the principal reasons: (a) memory limitations did not allow to explore a number of hyper parameters such as using all available classes in the dataset, or different sliding cube sizes. In both cases the resulting matrices were too large and the GPU would fail to handle them; (b) limiting the number of classes forced to place a great number of elements under the *other* class, which as a result created a class with low discriminative power due to the resulting amorphous shape and geometry, but highly represented in the dataset due to the number of voxels falling in this category; (c) using a generic constant size of the cube did not permit to capture the geometry of other elements; and (d) there was a lack of context regarding the content of the voxelized input with respect to the rest of the point cloud. An example of the training loss can be seen in Figure 3.

Adding Context: Following the previous failed attempt to learn space-semantics, my next step was to add global context as a second input to the network. Following the architecture described above, the network continued not to be able to learn. Once again memory limitations restricted the number of layers, number of filters, and other network parameters. The training loss of this network is marked with green line in Figure 4. Although the results are not as expected (see Figure 5), it did perform better than the previous network, which demonstrates that the global information was helpful, however not powerful enough to solve the ill-posed problem the sliding window approach created. I also experimented with the same architecture as that proposed in [6], however the results did not differ.

Enclosed Spaces: For the enclosed spaces approach I experimented with 5 and 10 semantic classes. In this case

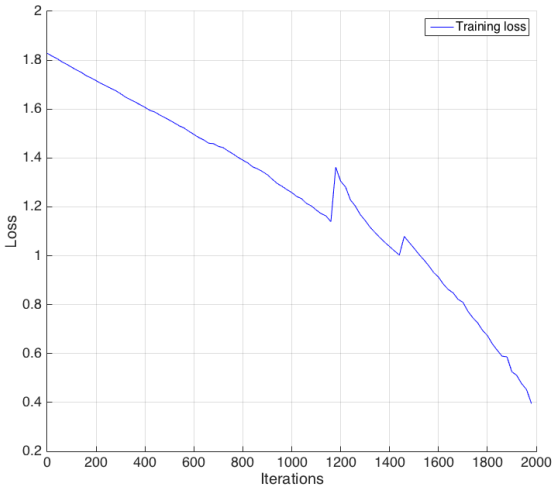

Figure 3. Training Loss of 3D Sliding Cube Approach.

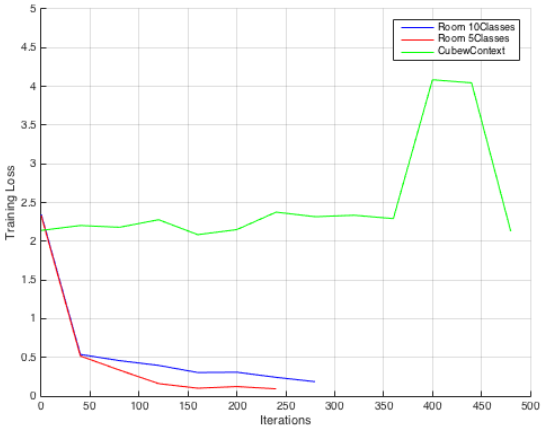

Figure 4. Training Loss.

Table 2. Mean testing accuracy of different apporaches

Approach	Sliding Cube+Context	Enclosed Space
Mean Acccuracy	0.05	0.60

the network was able to learn (see Figure 4), lines blue and red). Due to the smaller amount of training data than in the other experiments, I reduced the number of iterations to avoid over-fitting. In both cases the network performance shows similarities, although the behavior when using 10 classes appears slightly more stable (Figure 5).

The mean accuracy of the of the networks is also presented in Table 2.

5. Conclusion

During this project I faced a lot of challenges related to the size of the data and the memory limitations. These factors hindered the process in a great degree. From the exper-

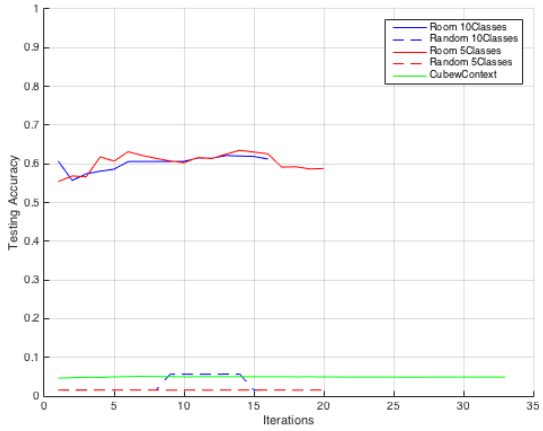

Figure 5. Testing Accuracy.

iments it became obvious that information about the context enabled the network to learn, from adding the global location of the sliding cube to providing whole space semantics. The final results of the enclosed space approach did boost the accuracy, however there are aspects of it that require improvement. First of all it assumes that one is able to identify rooms and consistently align them in a canonical reference coordinate system. Second, by normalizing spaces of different sizes into a unit cube, the objects in the scene undergo distortion. Although this step allows to convert *stuff* into *things* since now their dimensions are more consistent among different spaces, it compromises objects with consistent dimensions to arbitrary ones, which makes the learning process more difficult. Third, the grid resolution of the voxelization that was used was quite coarse, especially in the case of larger spaces.

A more sophisticated method needs to be used in order to achieve an end-to-end 3D Fully Convolutional approach for the problem of understanding space semantics, by *e.g.* combining segmentation, object proposal [25] and object detection approaches. This would allow to address both amorphous and not objects. Another interesting aspect would be to incorporate RGB information and observe how it affects the performance. However, this would essentially mean adding 3 more channels to each input and thus going back to the issue of memory limitations.

References

[1] Matterport 3d models of interior spaces. `http://matterport.com/`. Accessed: 2016-13-03.

[2] Theano theano 0.7 documentation. `http://deeplearning.net/software/theano/`. Accessed: 2016-13-03.

[3] U. E. P. Agency and the United States Consumer Product Safety Commission.

[4] L. A. Alexandre. 3d object recognition using convolutional neural networks with transfer learning between input channels. In *Intelligent Autonomous Systems 13*, pages 889–898. Springer, 2016.

[5] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[6] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[8] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Computer Vision–ECCV 2014*, pages 345–360. Springer, 2014.

[9] A. Hermans, G. Floros, and B. Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2631–2638. IEEE, 2014.

[10] N. Höft, H. Schulz, and S. Behnke. Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks. In *KI 2014: Advances in Artificial Intelligence*, pages 80–85. Springer, 2014.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, 2013.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[13] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2016.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *NIPS*, pages 244–252, 2011.

[16] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014.

[17] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *IJRR*, 2015.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, Nov. 2015.

[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 1, 2013.

[20] 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*, September 2015.

[21] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012.

[22] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[23] S. Ochmann, R. Vock, R. Wessel, M. Tamke, and R. Klein. Automatic generation of structural building descriptions from 3d point cloud scans. In *GRAPP 2014 - International Conference on Computer Graphics Theory and Applications*. SCITEPRESS, Jan. 2014.

[24] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2027–2034. IEEE, 2013.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[26] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[28] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012.

[29] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608. IEEE, 2011.

[30] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012.

[31] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[32] 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.