Point Cloud Generation from a Single 2D Image

Can Guemeli, Tymur Mirlas
Faculty of Mathematics and Computer Science, Technical University of Munich
Boltzmannstrasse 3, Garching near Munich, Germany
can.guemeli@tum.de, tymur.mirlas@tum.de

Abstract

In this project, we tackled with the problem of generating 3D point clouds from a single 2D image. We posed the problem as a supervised learning task, using RGB renderings and point clouds in ModelNet10 CAD models. We designed and trained a novel convolutional encoder-decoder architecture end-to-end. Our model slightly outperformed the strong baseline of Point Set Generation Networks, while having significant architectural advantages in terms of interpretability and adaptability. In this report, we describe our architecture, present qualitative and quantitative results of our experiments, and discuss our design decisions in the light of the recent research in point cloud generation. Moreover, we study some interesting characteristics of Chamfer Distance loss function used very commonly in our task, by trying various output layers, data sizes and semantic label varieties.

1. Introduction

3D point clouds allow representing the geometry of a scene in a compact form. Generating point clouds from 2D images will enable future research to utilize 3D geometry in many image analysis challenges, ranging from scene retrieval to object detection.

1.1. Contributions

In this work, we propose a novel architecture that uses multi-branch prediction with independent spatial transformer attentions. While trainable end-to-end using loss functions depicted in [2], our model can also be extended to more complex training schemes such as in [5] by allocating different branches to different tasks.

Following the optimization challenges we faced, we also include an ablation study on difficulty of optimizing the Chamfer Distance loss function in singe sample and single class settings, and we show the effect of different final layers on qualitative results obtained by single-sample optimization.

1.2. Problem Formulation

Given a 3-channel RGB image I_{RGB}, we predict a set of 3D coordinates $\hat{S}$ of size N, using a parametric function approximator h_θ. In our case, h_θ is a deep neural network, N and the resolution of I_{RGB} are known apriori.

$$\hat{S} = h_\theta(I_{RGB}) = \{(x_i, y_i, z_i)\}_{i=1}^{N} \quad (1)$$

We optimize parameters θ using a training set consists of image and ground-truth point set S pairs. We minimize the sum of a differential distance function d that compares $\hat{S}$ and S. In this work, we use Chamfer Distance as our distance function, defined as follows:

$$d(\hat{S}, S) = \sum_{p \epsilon \hat{S}} min_{q \epsilon S} ||p - q|| + \sum_{p \epsilon S} min_{q \epsilon \hat{S}} ||p - q|| \quad (2)$$

This distance can be seen as a bidirectional nearest neighbor loss.

1.3. Related work

For several years, deep point cloud processing has been extensively studied in 3D visual recognition tasks such as classification, segmentation and retrieval [6, 8, 1]. However, there are very few influential works on point cloud generation.

The earliest work we could track is the Point Set Generation Network [2]. This model first encodes the image using a convolutional network, and then predict point clouds by concatenating outputs produced by a transposed-convolutional network and a large fully connected network. Similar to our model, training is performed by optimizing Chamfer Distance end-to-end.

Another line of work involves using additional supervision by rendering point cloud targets and estimate depth maps that can be transformed into point clouds [5]. While this approach can be used to replace Chamfer Distance with more efficient loss functions, they also add more complexity to the dataset preparation process.

Our architecture adapts the simplicity of Point Set Generation Network, while its multi-branch nature makes it adaptable to grid-based methods.

2. Dataset

We use CAD models from the ModelNet10 dataset [9], using the 12-view renderings depicted in [7]. Renderings are obtained from the GitHub repository https://github.com/jongchyisu/mvcnn_pytorch.

ModelNet10 dataset consists of .off files that contain a set of points and triplets of points that form triangles. We sampled a fixed number of points as the point cloud target for each image, normalized the targets to the range $[-1, 1]$, and rotated them to match the viewpoint of the rendering. We utilized data processing modules available in the PyTorch Geometric library [3] for the aforementioned preprocessing steps.

3. Architecture

Following the [2], we wanted to adopt a simple architecture that consists of an image encoder using convolutions and a point cloud decoder using transposed convolutions. Different from dense image prediction tasks such as semantic segmentation and depth estimation, the number of output units our decoder has to predict is much less than the number of pixels in the input image. In particular, for a 224×224 image, we want to predict around 1024 points which correspond to 32×32 grid with x, y, z channels. This is an issue because we want to use both high-resolution and low-level features and high-level low-resolution features from the image encoder.

To solve this problem, we divided the decoder into K transposed convolutional branches, each applies a spatial transformer attention [4] to relevant feature maps in the encoder network. We use $K = 6$ and make each branch to output a 14×14 grid with x, y, z channels. The output of each branch is then added to a regular 2D grid as in [5]. Set union of all these branches are used as prediction and optimized via Chamfer Distance.

3.1. Advantages over the Point Set Generation Networks

To solve the resolution problem discussed above, Point Set Generation Network stacks two encoder-decoder networks in their best-performing version. Although this method allows them to use features extracted from different resolutions, it is hard to interpret how this architecture works. In comparison, our architecture allows different branches to attend different parts of the image to predict corresponding subsets of point clouds, therefore is more interpretable. Moreover, our architecture is more refined in the sense that it consists only of single encoder and decoder.

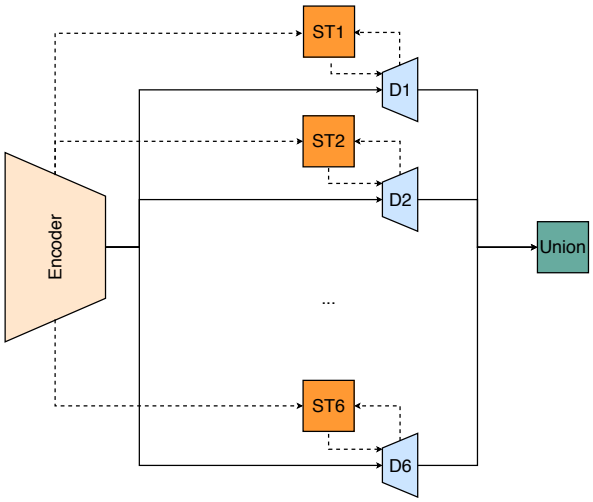

Figure 1. Multi-branch CNN for point cloud generation.

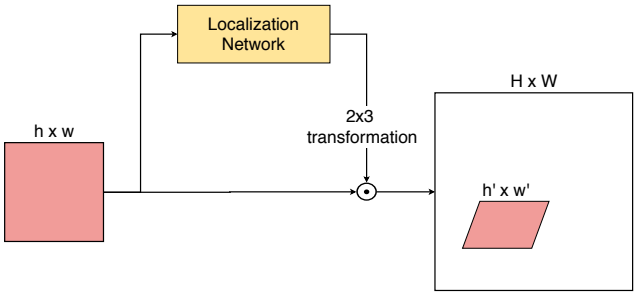

Figure 2. Spatial Transformer Network as feature pooling module.

4. Results

4.1. Final Layer Ablation Study

In this section, we explored the effect of different output layers to prediction quality. For any neural network prediction $\hat{S}_{raw} = h_\theta(I_{RGB})$, we experimented with grid deformation, tanh, and linear applied . These output layers are as follows, respectively:

$$\hat{S} = \hat{S}_{raw} + [X, Y, 0] \tag{3}$$

$$\hat{S} = tanh(\hat{S}_{raw}) \tag{4}$$

$$\hat{S} = \hat{S}_{raw} \tag{5}$$

where $[X, Y] = meshgrid(-1, 1)$.

We compare the results in single sample overfitting and one-class training.

4.1.1 Overfitting to one sample

We observed convergence to a local minimum when trying to overfit to a single sample. We observe grid deformation leads to a higher quality optimization process and details by more evenly distributing the points.

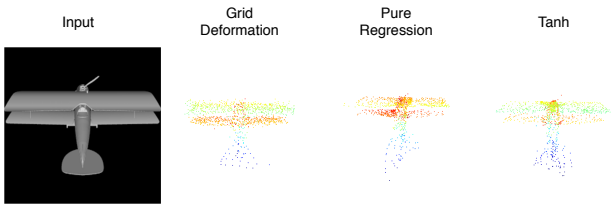

Figure 3. Overfitting results on 1 sample for Grid Deformation, Linear Transformation and Tanh final layers.

4.1.2 Training on a Single Class

Here, we see the output layer doesn't have a significant effect, qualitatively or quantitatively.

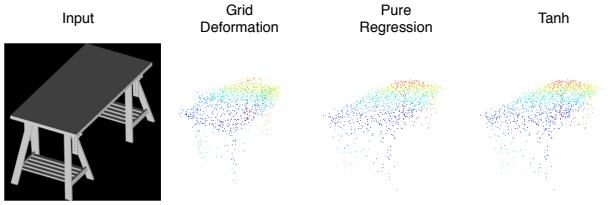

Figure 4. Training results for a class table for Grid Deformation, Linear Transformation and Tanh final layers.

4.2. ModelNet10 Training Results

We obtained a stable training schedule when training our model. We used learning rate 1e-4, weight decay of 1e-4, batch size 16 and Adam Optimizer.

We reimplemented the baseline Point Set Generation architecture in PyTorch, adapting the dimensionality to our dataset and framework. We outperformed the baseline architecture slightly on average and in most classes when the same training procedure is applied with the same hyperparameters.

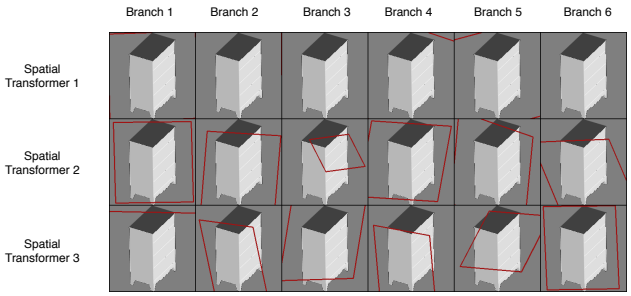

Figure 6. Visualization of the attention layers. One can see that different attention layers attent different regions of the input.

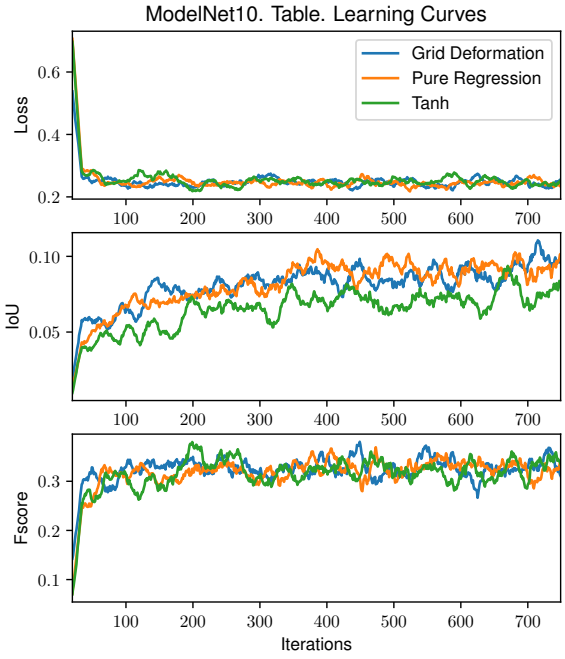

Figure 5. Learning curves for a validation set of a class "table" for Grid Deformation, Linear Transformation and Tanh final layers.

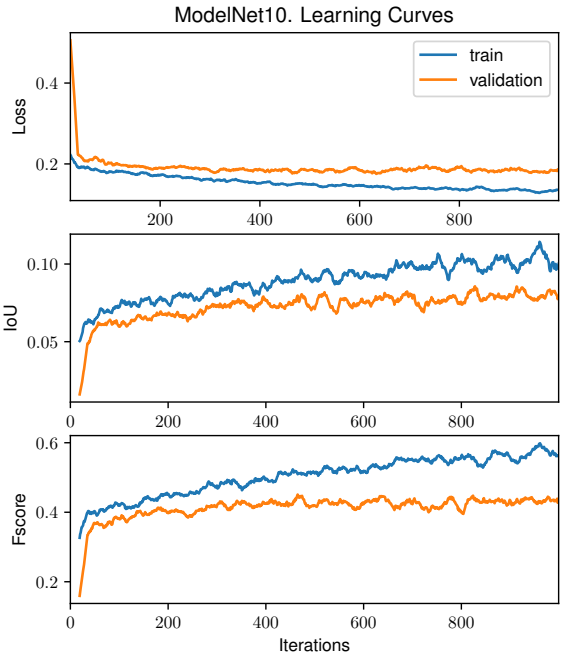

Figure 7. Learning curves for train and validation sets of the ModelNet10.

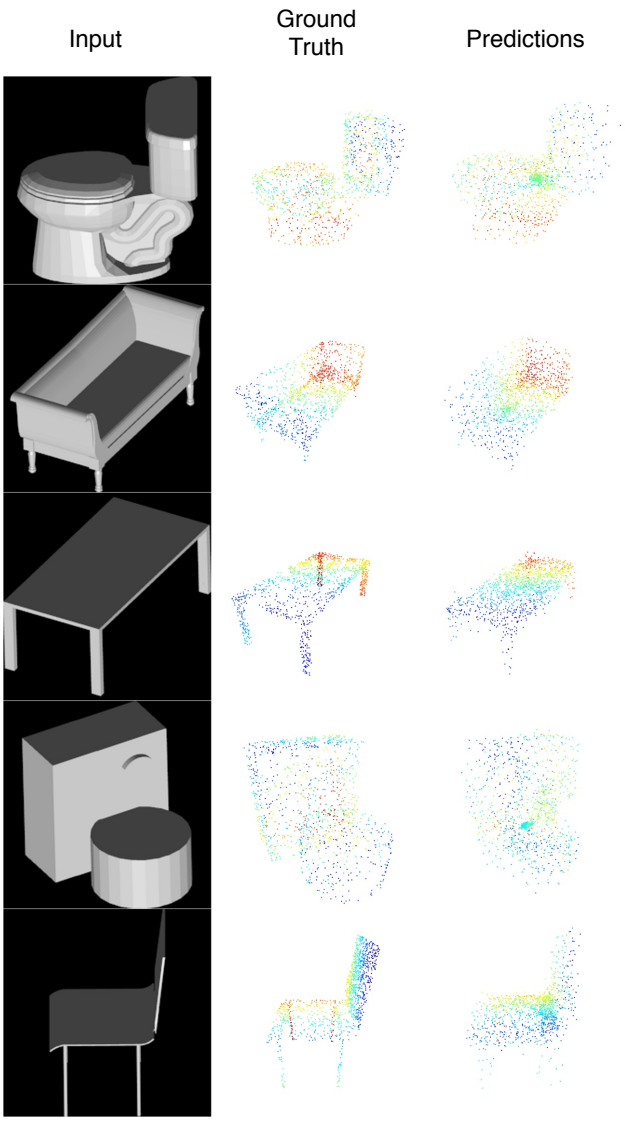

Figure 8. Visualization of the network outputs.

	Loss		Fscore		IoU	
	Ours	**PSG**	**Ours**	**PSG**	**Ours**	**PSG**
bathtub	**0.172**	0.174	**0.462**	0.446	**0.086**	0.083
bed	**0.150**	0.154	**0.533**	0.509	**0.088**	0.085
chair	**0.173**	0.175	**0.466**	0.444	**0.097**	0.090
desk	**0.214**	0.218	**0.370**	0.345	**0.058**	0.055
dresser	**0.210**	0.214	**0.300**	0.284	**0.040**	0.036
monitor	0.154	**0.153**	**0.550**	0.538	**0.108**	0.101
night stand	0.245	**0.245**	**0.256**	0.239	0.038	**0.038**
sofa	**0.144**	0.147	**0.526**	0.514	**0.103**	0.100
table	**0.226**	0.237	**0.351**	0.323	0.085	**0.088**
toilet	**0.161**	0.165	**0.462**	0.430	**0.082**	0.074
Average	**0.185**	0.188	**0.428**	0.407	**0.078**	0.075

Table 1. Quantitative comparison of our model vs. baseline

5. Conclusion

During the ablation study, we have checked the hypothesis, that final layer of the network affects its performance. As a conclusion, we can say, that final layer indeed matters only in case of overfitting to one sample, and does not matter in case of large-scale training of deep networks.

As the baseline, we have the Point Set Generation Network, which utilizes convolutional layers to regress point clouds. We have reached the state of the art performance in comparison with the baseline. Moreover, our model has more capacity, is more interpretable, and scales better for architectural enhancements.

References

[1] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4470–4479, 2018.

[2] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[3] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[4] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[5] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[7] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik 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.

[8] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[9] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong 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.