A Network Architecture for Point Cloud Classification via Automatic Depth Images Generation - Supplementary Material

Riccardo Roveri[1†], Lukas Rahmann[1†], A. Cengiz Öztireli[2*], Markus Gross[1]
[1]Department of Computer Science, ETH Zürich
[2]Disney Research Zürich
[†]Equal contribution from both authors
{rroveri, grossm}@inf.ethz.ch, lrahmann@student.ethz.ch, cengiz.oztireli@disneyresearch.com

1. View Selection for Our 2 Views Architecture

Similarly to Figure 6 of our paper, we show in Figure 1 (top) the density of the learned views for six testing models, this time for the case of our 2 views architecture. The first row is the density of the first view, and the second row the one of the second view. Again, the density functions contain peaks and further regions with very low values, optimized for different objects. Most interestingly, one can also notice how the density functions of the two views are complementary to each other: the regions of high values in the first view usually have low values in the second view, and the other way around. This demonstrates how our network chooses different view points to combine their features and optimize the results.

Like in our paper, for each test point cloud, we sample the views corresponding to the highest value of the views densities (i.e. the most likely views estimated by our network), and show the depth images generated by our network for those views in Figure 1 (bottom, first row for the first view and second row for the second view). It is noticeable how the views complement each other, by showing different features of the objects. For example, one can see the side of the cone and its bottom part, the side of the cup and its inside, the top of the bench and its bottom part.

Like in Figure 7 of our paper, in Figure 2 we show further view density functions and depth images for the highest probability views, for a class of objects (lamps) using our 2 views architecture. Again, the pairs of learned views are similar for objects of the same class, demonstrating how our network specializes to the different objects. It is also clear how the views are complementary to each other. In particular, the first view tends to show the lamps from the side, and the second view often adds a shifted view point.

All the view density functions in the paper and in the Supplementary Material are spheres rendered from a fixed arbitrary point view.

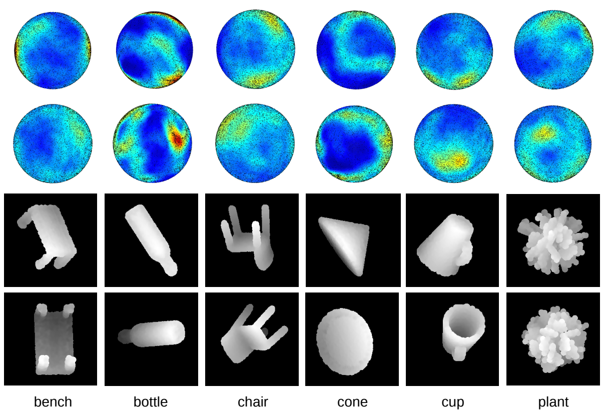

Figure 1. Learned view density functions (top, first row for the first view and second row for the second view), depth images generated by our network corresponding to the most likely learned views (bottom, first row for the first view and second row for the second view), for our 2 views architecture.

2. Comparison with PCA

In Figure 3 we show a comparison of our selected views to views selected by exploiting the PCA components, as explained in the original paper. The three objects are indistinguishable from the PCA views, while easily recognizable with our method. Both the generated depth images and the point clouds rendered from the view points are shown.

3. Failure Cases and Comparison with Meshes

Due to the low resolution of the input point cloud and the ambiguity of some models in the dataset, there exists a subset of classes which are hard to classify. In Figure 4, three objects from these classes are shown. Like PointNet and the other most recent papers that handle point clouds, our method fails to properly classify them, as the correspondent

*Work done while at ETH Zürich

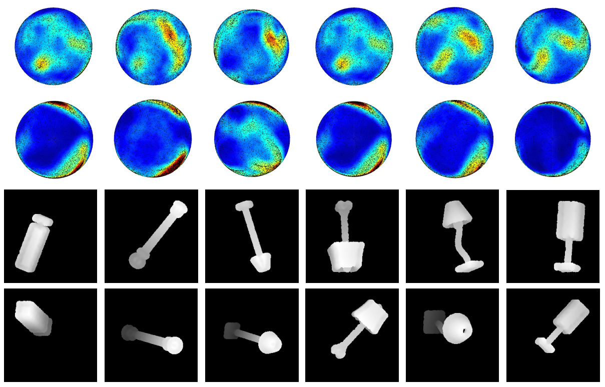

Figure 2. Learned view density functions and the depth images corresponding to the views with highest probability for a class of objects (lamps), for our 2 views architecture. The first row of the density functions and of the depth images corresponds to the first view, and the second row to the second view.

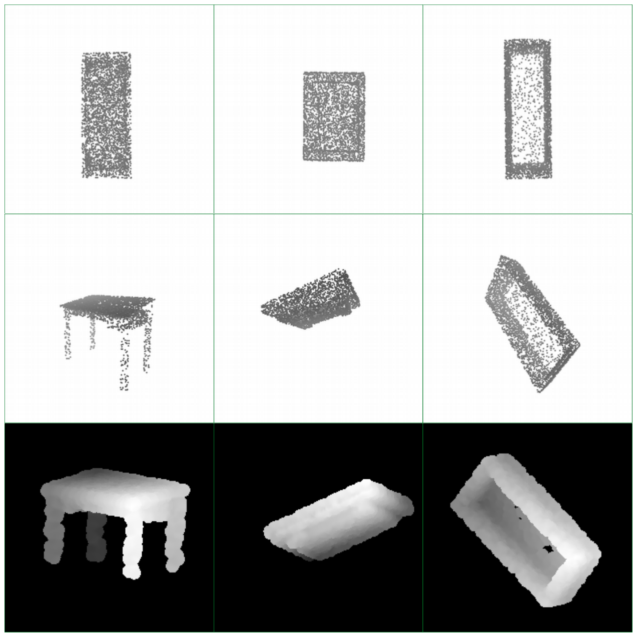

Figure 3. The views selected by PCA (top: point clouds), and by our method (middle: point clouds, bottom: depth images).

generated depth images are ambiguous. The very detailed meshes used in MVCNN [1] include more features (because of their high resolution), and we believe this is the reason for their higher accuracy numbers compared to point-based methods. In Figure 5, we compare the original high resolution meshes of two models with the meshes obtained by reconstructing a point cloud of the objects sampled with the same number of points that we use (2048), using the commonly adopted Poisson reconstruction method with optimal parameters. As one can see, the reconstructed meshes (on the right) contain way less details (XBox), and, sometimes, the reconstruction even fails to correctly represent the shape of the object (plant). This shows how generating meshes from our sparse point clouds would not facilitate the problem.

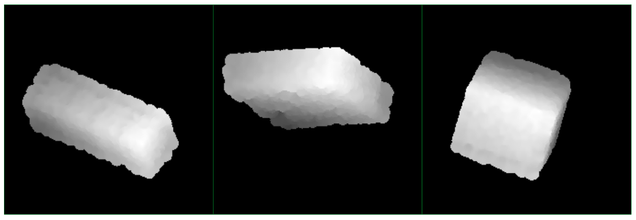

Figure 4. Three classes where our generated views are ambiguous.

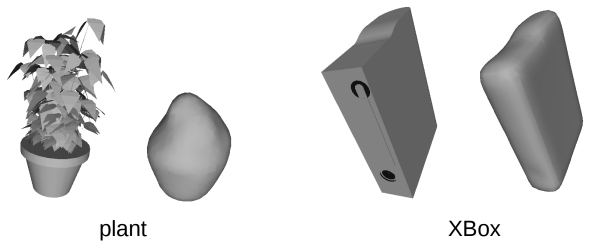

Figure 5. The original high resolution mesh (left) and the reconstructed mesh from a point cloud of 2048 points (right), for two models (plant and XBox).

4. View Selection Convergence Examples

In the accompanying video, we show how the generated view by our single view architecture for three testing objects (cup, couch and night stand) changes at every training step. The objects were fed to the network with the same rotation at every step, and the generated depth images were used as frames to compose the video. Since we gradually decrease the learning rate during training, it is expected to slowly see convergence to a certain view and see the flickering in the video decrease. It is noticeable, more importantly, how all the three final views are proper and useful for the different objects. In addition, one can also observe how convergence happens at different training time for the three objects (first for the cup, then for the couch and finally for the more difficult night stand). This further proves how our view selection component specializes its behavior depending on the structure of the objects.

5. Comparison with Random Views Alternative

In Table 1, Table 2, and Table 3, we present the difference of obtained accuracy between our original method and the random views alternative ($\pm$ Acc.), for 1, 2 and 4 views respectively, for the classes where the absolute difference was at least 1%. As for the results in the paper, the numbers from 5 evaluations were averaged for each case. The classes in bold are the ones which were always better classified with the random alternative (bench and radio), or always better classified with our original architecture (piano, vase, desk, table, baththub, dresser and night stand). It is noticeable how our original architecture performs consistently better in more classes than the random alternative, due to its specialized learned views that adapt to classes. As explained in the paper, we believe that the random alternative can sometimes perform better than our method due to hard, ambiguous classes present in the dataset (e.g., radio that can be confused with other regular objects).

6. Gradients for Depth Image Generation

In this section, we present the gradients of our custom module for depth images generation.

We name the nominator and denominator of the function $f(c)$ of Equation 4 in the original paper as follows:

$$f(c) = \frac{\sum_{p \in P''(c)} g(c,p) p_z}{\sum_{p \in P''(c)} g(c,p)} = \frac{f_c^1(P)}{f_c^2(P)} \tag{1}$$

The gradients with respect to the position of a point $p \in P$ for f_1 can be defined as:

$$\frac{\partial f_c^1(P)}{\partial p_z} = \begin{cases} 0 & if \quad p \notin P''(c) \\ g((c_x, c_y), (p_x, p_y)) & otherwise \end{cases} \tag{2}$$

$$\frac{\partial f_c^1(P)}{\partial p_y} = \begin{cases} 0 & if \quad p \notin P''(c) \\ \frac{(c_y - p_y)}{\sigma^2} g((c_x, c_y), (p_x, p_y)) p_z & otherwise \end{cases} \tag{3}$$

The gradients for f_2 are defined as follow:

$$\frac{\partial f_c^2(P)}{\partial p_z} = 0 \tag{4}$$

$$\frac{\partial f_c^2(P)}{\partial p_y} = \begin{cases} 0 & if \quad p \notin P''(c) \\ \frac{(c_y - p_y)}{\sigma^2} g((c_x, c_y), (p_x, p_y)) & otherwise \end{cases} \tag{5}$$

Similarly, the gradients with respect to p_x can be computed.

The final derivative for $f(c)$ is then defined by utilizing the chain rule.

References

[1] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015. 2

± Acc.	-0.06	-0.03	-0.03	-0.02	-0.02	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	-0.01	-0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.03	0.03	0.03	0.04	0.05
	bowl	car	person	**bench**	**radio**	range hood	laptop	stairs	cone	sink	cup	curtain	glass box	**piano**	**vase**	toilet	sofa	bookshelf	bed	tv stand	lamp	**desk**	door	**table**	plant	keyboard	**bathtub**	**dresser**	**night stand**	stool	flower pot

Table 1. Difference of obtained accuracy between our original method and the random views alternative (± Acc.), for 1 view, per class.

± Acc.	-0.05	-0.04	-0.04	-0.02	-0.01	-0.01	-0.01	-0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03	0.04	0.04	0.05	0.06	0.06	0.07	0.07
	wardrobe	stairs	**radio**	laptop	flower pot	chair	plant	**bench**	glass box	bed	mantel	**piano**	tent	**night stand**	tv stand	**dresser**	range hood	**vase**	sofa	cone	door	person	cup	sink	bowl	curtain	**table**	stool	**desk**	xbox	**bathtub**

Table 2. Difference of obtained accuracy between our original method and the random views alternative (± Acc.), for 2 views, per class.

± Acc.	-0.06	-0.04	-0.03	-0.03	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.03	0.03	0.04	0.04	0.04	0.04
	tent	wardrobe	door	**bench**	plant	cup	tv stand	curtain	keyboard	bed	bowl	stool	glass box	bottle	monitor	**radio**	car	**table**	**piano**	**desk**	chair	mantel	**vase**	**night stand**	cone	lamp	person	sink	stairs	flower pot	xbox	**dresser**	**bathtub**

Table 3. Difference of obtained accuracy between our original method and the random views alternative (± Acc.), for 4 views, per class.