3D Reconstruction from Single Image

John Penington
University of Michigan
jaldenp@umich.edu

Yu Mei
University of Michigan
ericmei@umich.edu

Zhengyu Huang
University of Michigan
zyhuang@umich.edu

Mohammadreza Tavasoli Naeini
University of Michigan
tavasoli@umich.edu

Abstract

We present an architecture in this paper to predict the 3D representation given a single image. The architecture is called the TL-embedding network and consists of an autoencoder and a CNN pretrained on the image net classification task. A small subset of the ShapeNet dataset is used to train and test the network, while we use precision to compute its effectiveness. The proposed network performs well on dense objects like beds and sofas but poorly on objects with sparse components like chairs. Our re-implementation of the network is compared to the original TL-Network using average precision. Our image net performance is close to the original image net, but our autoencoder performance is worse than the original. However, we are confident that we have implemented the model correctly and it will have a better performance with a larger training dataset.

1. Introduction

Learning to perceive the 3D world is a challenging problem and is a classical computer vision problem. One goal of this problem is to reconstruct a 3D model of the object, given one input image. This is a challenging problem, especially in the case of single input image. As there are regions of the object that are occluded from the view point, reconstructing these occluded regions requires good prior information about the object. In this project, we aimed at reconstructing the 3D object model, given a single input image, by following the approach proposed by Girdhar et al.[4]. During training, we use an 3D autoencoder to encode the representation of the 3D object into a feature vector. At the same time, we have another network learn to regress from the input image to the same feature vector. This ensures the feature vector is predictable from the input image and also is generative in the sense that it can be fed into the decoder and reconstruct the 3D shape. As a result, at the test time, we can estimate a good feature vector of the object encoding its 3D shape using a single input image and use it to reconstruct the object using the trained decoder.

2. Related work

Choy [2] takes one or multiple input images and uses a 3D recurrent neural network (3D-R2N2) to reconstruct the 3D shape. Recently, Mildenhall et al. [6] proposed to use a fully connected neural network to model a continuous scene and trains the network by minimizing the error between the rendered image and the observed input image. [3] used point cloud to reconstruct 3d shape from the single 2D input object. [10] proposed the Encoder-Decoder network and defined projection loss, which interacts between 3d and 2d representation to reconstruct the 3D shape of the 2D image without annotation. [8] proposed deep generative models of 3D structures. By probabilistic inference, [8] recover 3D shape from 2D images. They did not use ground truth labels in their generative and inference networks. [9]proposed a model that learns scene representation as well as 3D and textual information by 3D scene de-rendering network. [9] offered an encoder-decoder network structure that can use for image reconstruction from scene representation and 3D-aware image editing.

3. Method

3.1. Network architecture

The problem we proposed to solve can be divided into two parts: (1) can we represent the 20 X 20 X 20 voxel grid with a lower dimension embedding feature vector and (2) can we reconstruct a 3D voxel grid with a single 2D image with the extracted features.

To solve the above issues, we propose a model following the network architecture in [4] and re-implement the TL-embedding network, as shown in Figure 1, where T refers to the training network and L refers to the testing network.

The T network consists of two parts: an autoencoder and an image net. The autoencoder has three components, an encoder with 4 convolution layers, a decoder with 5 deconvolution layers and a 64D full-connected linear layer in between. Each convolution layer are connected with a Parametric Rectified Linear Unit (PReLU). The 3D voxel maps are fed into the autoencoder and downsampled into a 64 features vector and then are upsampled by the decoder to reconstruct the original voxel grid. On the top of the autoencoder, we add a Sigmoid layer to scale the output of the autoencoder to [0,1], representing the confidence of a voxel at a certain location.

After examining different CNN for feature extraction (i.e. ResNet), we will use the pre-trained AlexNet [5] as our backbone image net for feature extraction, as proposed in [4], because most of these CNN yield very closed final results. The original AlexNet is an image classifier for 1000 classes with 5 convolution layers and 2 dense layers. To match the dimension of the embedding vector, we add another 64D full-connected linear layer on the top of the AlexNet. To prevent the overfitting, we will freeze the 3 lowest convolution layer in the AlexNet. The core idea of our model is that the image net are expected to extract the same feature vector from a single 2D image as the embedding vector from its corresponding 3D voxel representation, such that the decoder can reconstruct the same predicted voxel maps.

In L network, we remove the encoder and connect the image net and the decoder. Hence, the feature vector extracted from the 2D RGB image by the image net are directly fed into the decoder for the reconstruction.

3.2. Data generation

We used the ShapeNet dataset [1] for the network training. ShapeNet dataset contains about 220K 3D models. For the training time consideration, we used a subset of common indoor objects data: chair (289 models), table (331 models), sofa (271 models), cabinet (232 models) and bed (233 models). We chose randomly 80% of the data as the training set and the remaining as the test set. For each model, we imported the 3D model into Blender and rendered 72 views of the object. We chose 3 elevation angles for the viewing camera (15°, 30° and 45°) and for each elevation angle, we render the image at 24 evenly spaced azimuth angles. This results in total 72 rendered images per 3D model. In order to prevent the network from overfitting the sharp edges when rendered on a plain background, following the approach of the original paper, we rendered the model on random background images from the indoor scene recognition dataset [7] containing 15620 images. The rendered images have spatial resolution of 227 X 227 and the voxel data used in the auto-encoder is down-sampled to a resolution of 20 X 20 X 20.

4. Experiments

4.1. Training

We have 2 training phases. In the phase one, we train the autoencoder independently with a collection of 3D voxel mapping as the ground truth. We take the end-to-end output of the autoencoder and train it with the Binary Cross Entropy loss with the form:

$$E = -\frac{1}{N}\sum_{n=1}^{N}[p_n log\hat{p} + (1-p_n)log(1-\hat{p}_n)]$$

where p_n is the ground truth voxel label such that $p_n \in \{0,1\}$ and $\hat{p}_n$ is the predicted probability such that $\hat{p}_n \in [0,1]$, and N is the number of voxels (i.e. 20^3 in our case). We apply a 10^{-2} learning rate with a Stochastic Gradient Descent (SGD) optimizer and train the autoencoder for 150 epochs.

In the phase two, we focus on the image net. We take a collection of image-voxel pair as input, where the voxel map is fed into the trained encoder from the first phase and the image is fed into the image net. We treat the 64D embedding vector from the encoder as ground truth and train the image net to regress the embedding vector with the Mean Squared Error loss (MSE) with the form:

$$E = \frac{1}{N}\sum_{n=1}^{N}(f_{Encoder_n} - f_{imagenet_n})^2$$

where $f_{Encoder_n}$ is the n^{th} value from the encoder's embedding vector and $f_{imagenet_n}$ is the n^{th} feature extracted from image net. N is the dimension of the embedding space (i.e. 64 in our case). As mentioned above, the 3 lowest layers of the image net will be frozen. We apply a 10^{-4} learning rate with a Adam optimizer and train the image net for 60 epochs.

The paper [4] proposes a third training phase where both the autoencoder and the image net are fine-tuned together with a joint loss. However, we decide to skip the third phase because it does not improve the performance significantly, as shown in [4], while it takes massive time to train with our single GPU setup.

We split the whole data set into training set and testing set to measure our model performance. Specially, 80% of the whole data set is used for training, while the rest 20% is used for testing. In both phases, we conduct a 3-fold cross-validation.

To improve the performance, we use a scheduler to dynamically adjust the learning rate. For phase one, we drop the learning rate by 90% for each 50 epochs (i.e. the learning rate becomes 10^{-3} at epoch 51 and 10^{-4} at epoch 101, etc). For phase two, we drop the learning rate by 90% for each 30 epochs.

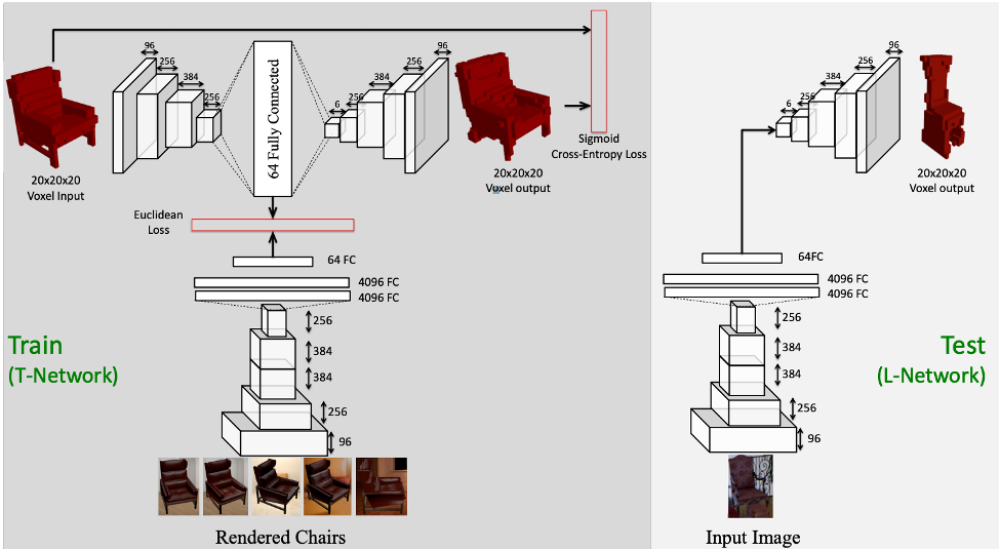

Figure 1. The TL-Network [4]

4.2. Testing

In testing phase, we remove the encoder network in the autoencoder and connect the image net with the decoder network. The input is a single RGB image fed in to the image net and we take the end-to-end output of the R network as our prediction. As the decoder is connected to the Sigmoid layer, the output is a 3D 20^3 array scale to [0,1] at each index. Since most of the space is empty, most of the output values are closed to zero. As the model is more sensitive to the positive number, we divide the interval for each index in the following, instead of simply split the interval by half:

- (0.0, 0.2): Not a voxel
- (0.2, 0.4): Edged cases
- (0.4, 0.6): Weak confidence a voxel presented
- (0.6, 0.8): Medium confidence a voxel presented
- (0.8, 1.0): Strong confidence a voxel presented

4.3. Qualitative results

The qualitative results are shown in Figures 2 and 3. The output has red, yellow, green, and white voxels. These colors corresond to the output intervals above. Red is (0.8, 1.0), yellow is (0.6, 0.8), green is (0.4, 0.6), and white is (0.2, 0.4).

The autoencoder reconstruction and image net prediction can be seen in Figure 2. Figure 2 contains a sample test output for the five furniture categories in our dataset. The image net comes close to predicting the same voxels as the autoencoder reconstruction, which is correct as the image net was trained using the encoder. The figure also shows that our network is good at predicting the core structure (as there are many red and yellow voxels in the middle), but is poor at predicting the edges (lots of white on edges). Figure 2 also hints at what our TL-Network is good at predicting dense objects like beds and sofas while poor at predicting objects with sparse components like chairs.

Figure 3 shows a couple of the failure points of our network. In the top prediction, the autoencoder fails to encode the sparse columns of the table so the image net (having been trained on the encoder which fails to encode columns) fails to predict them as well. In the bottom prediction, the chair texture is very similar to the background. This results in a very poor autoencoder reconstruction and image net prediction.

4.4. Quantitative results

	Chair	Table	Sofa	Cabinet	Bed	Average
$n = 100$	57.4	49.1	81.9	81.0	72.1	68.3
$n = 600$	71.1	69.2	93.4	92.9	82.2	81.8
Girdhar	96.4	97.1	99.1	99.3	93.8	97.6

Table 1. Average Precision for Voxel Reconstruction

	Chair	Table	Sofa	Cabinet	Bed	Average
$n = 100$	29.5	17.5	47.0	57.3	35.0	37.2
$n = 600$	45.0	41.6	74.3	77.4	59.6	59.7
Girdhar	66.9	59.7	79.3	79.3	41.9	65.4

Table 2. Average Precision for Voxel Prediction

Table 1 and 2 above describe the quantitative results. As mentioned in 3.2, we generated a subset from ShapeNet including chairs (289 models), table (331 models), sofa (271

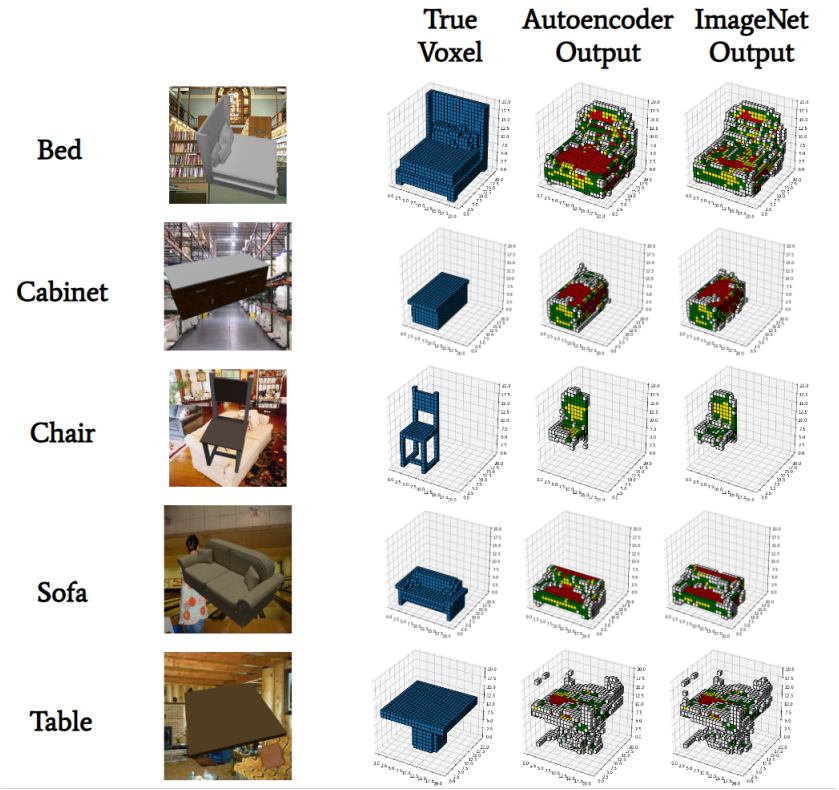

Figure 2. Reconstruction and prediction results from CAD test dataset for each category

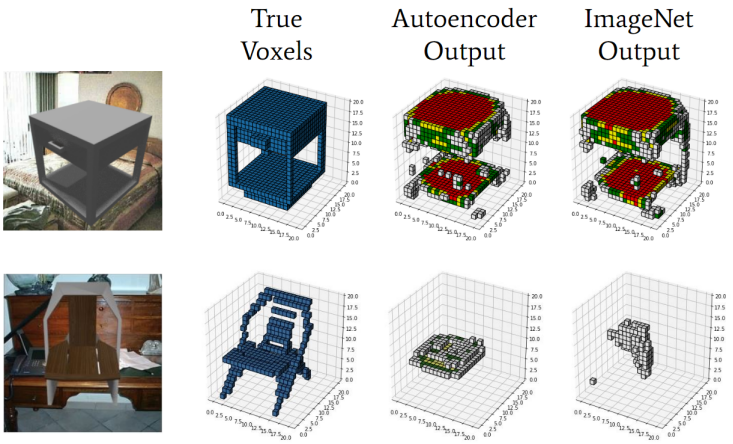

Figure 3. Examples of poor prediction

models), cabinet (232 models) and bed (233 models). However, due to the time and resource restriction, we didn't utilize all data we have generated. Instead, we trained our model with two different size of dataset. The first model (labeled as $n = 100$) is trained with 100 voxel samples randomly selected from the samples pool and 7200 total generated images, while the second model (labeled as $n = 600$) is trained with 600 voxel samples and 43200 total images. The Girdhar row contains the results from the [4], which contains the original TL-Network we re-implemented.

We are using average precision to evaluate our results because it is a better metric in this case than total accuracy. Voxel representations are sparse so a high accuracy score can be achieved by predicting a lot of zeros. Average preci-

sion is used because it is the total number of correctly filled in voxels divided by the total number of filled voxels. This rewards the models that can better reconstruct the object in a sparse space.

As discussed in 4.3, Tables 1 and 2 show our TL-Network does better on dense objects like sofas and cabinets and worse on object with sparse components like chairs and tables. Our average precision ($n = 600$) for both the voxel reconstruction (autoencoder) and prediction (image net) is worse than Girdhar's results. There are a few reasons for this discrepancy. The first one is that Girdhar trained with 16288 voxel samples while we trained with 600. We only had one GPU to train the model so we could not use a lot of training samples, as it would have taken weeks for us to train with that amount of data. The reconstruction precision is further from Girdhar's result as we trained it with less than the image net. For each voxel, the image net was trained with 72 rendered images while the autoencoder was only trained with the voxel. In both tables, we show that more training data results in a better precision score. The precision with $n = 600$ samples is better in every category over $n = 100$ samples. For this reason, we believe our model would perform closer to Girdhar's results with more training data.

5. Conclusions

Our PyTorch implementation of the TL-embedding network was able to predict the core structure of dense objects (e.g. beds and sofas) but struggled with the edges and sparse objects (e.g. legs of chairs and columns). While we were forced to use a much smaller dataset than the original TL-Network [4] due to fewer computational resources, our image net implementation was close to the original's results. On the other hand, the small dataset caused our autoencoder to preform significantly worse than the original's. To fully re-implement the TL-Network, more computationally effective GPUs and a larger training set are required.

References

[1] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[2] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[3] 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.

[4] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Gupta Abhinav. Learning a predictable and generative vector representation for objects. *Computer Vision – ECCV 2016*, Aug 2016.

[5] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 2012.

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[7] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

[8] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in neural information processing systems*, pages 4996–5004, 2016.

[9] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pages 540–550, 2017.

[10] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in neural information processing systems*, pages 1696–1704, 2016.