Learning a Predictable and Generative Vector Representation for Objects

Rohit Girdhar[1] David F. Fouhey[1] Mikel Rodriguez[2] Abhinav Gupta[1]

[1]Robotics Institute, Carnegie Mellon University [2]MITRE Corporation
{rgirdhar,dfouhey,abhinavg}@cs.cmu.edu, mikel@cs.ucf.edu

Abstract. What is a good vector representation of an object? We believe that it should be generative in 3D, in the sense that it can produce new 3D objects; as well as be predictable from 2D, in the sense that it can be perceived from 2D images. We propose a novel architecture, called the TL-embedding network, to learn an embedding space with these properties. The network consists of two components: (a) an autoencoder that ensures the representation is generative; and (b) a convolutional network that ensures the representation is predictable. This enables tackling a number of tasks including voxel prediction from 2D images and 3D model retrieval. Extensive experimental analysis demonstrates the usefulness and versatility of this embedding.

1 Introduction

What is a good vector representation for objects? On the one hand, there has been a great deal of work on discriminative models such as ConvNets [18, 32] mapping 2D pixels to semantic labels. This approach, while useful for distinguishing between classes given an image, has two major shortcomings: the learned representations do not necessarily incorporate the 3D properties of the objects and none of the approaches have shown strong generative capabilities. On the other hand, there is an alternate line of work focusing on learning to generate objects using 3D CAD models and deconvolutional networks [5, 19]. In contrast to the purely discriminative paradigm, these approaches explicitly address the 3D nature of objects and have shown success in generative tasks; however, they offer no guarantees that their representations can be inferred from images and accordingly have not been shown to be useful for natural image tasks. In this paper, we propose to unify these two threads of research together and propose a new vector representation (embedding) of objects.

We believe that an object representation must satisfy two criteria. Firstly, it must be **generative in 3D**: we should be able to reconstruct objects in 3D from it. Secondly, it must be **predictable from 2D**: we should be able to easily infer this representation from images. These criteria are often at odds with each other: modeling occluded voxels in 3D is useful for generating objects but very difficult to predict from an image. Thus, optimizing for only one criterion, as in most past work, tends not to obtain the other. In contrast, we propose a novel architecture,

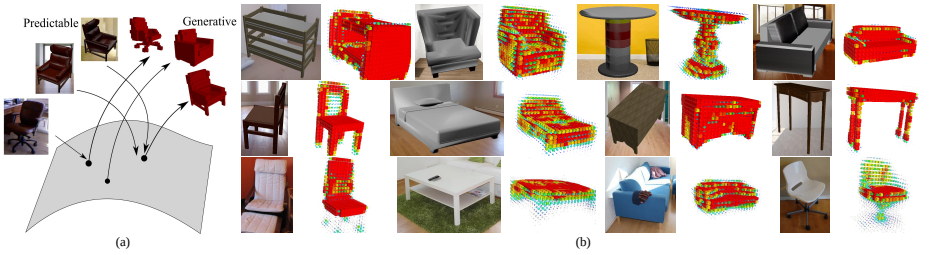


Fig. 1: (a) We learn an embedding space that has generative capabilities to construct 3D structures, while being predictable from RGB images. (b) Our final model's 3D reconstruction results on natural and synthetic test images.

the TL-embedding network, that directly optimizes for *both* criteria. We achieve this by building an architecture that has two major components, joined via a 64-dimensional (64D) vector embedding space: (1) An autoencoder network which maps a 3D voxel grid to the 64D embedding space, and decodes it back to a voxel grid; and (2) A discriminatively trained ConvNet that maps a 2D image to the 64D embedding space. By themselves, these represent generative and predictable criteria; by joining them, we can learn a representation that optimizes both.

At training time, we take the 3D voxel map of a CAD model as well as its 2D rendered image and jointly optimize the components. The auto-encoder aims to reconstruct the voxel grid and the ConvNet aims to predict the intermediate embedding. The TL-network can be thought of as a 3D auto-encoder that tries to ensure that the 3D representation can be predicted from a 2D rendered image. At test time, we can use the autoencoder and the ConvNet to obtain a representation for 3D voxels and images respectively in the common latent space. This enables us to tackle a variety of tasks at the intersection of 2D and 3D.

We demonstrate the nature of our learned embedding in a series of experiments on both CAD model data and natural images gathered in-the-wild. Our experiments demonstrate that: (1) our representation is indeed generative in 3D, permitting reconstruction of novel CAD models; (2) our representation is predictable from 2D, allowing us to predict the full 3D voxels of an object from an image (an extremely difficult task), as well as do fast CAD model retrieval from a natural image; and (3) that the learned space has a number of good properties, such as being smooth, carrying class-discriminative information, and allowing vector arithmetic. In the process, we show the importance of our design decisions, and the value of joining the generative and predictive approaches.

2 Related Work

Our work aims to produce a representation that is generative in 3D and predictable from 2D and thus touches on two long-standing and important questions in computer vision: how do we represent 3D objects in a vector space and how do we recognize this representation in images?

Learning an embedding, or vector representation of visual objects is a well studied problem in computer vision. In the seminal work of Olshausen and Field [26], the objective was to obtain a representation that was sparse and could reconstruct the pixels. Since then, there has been a lot of work in this reconstructive vein. For a long time, researchers focused on techniques such as stacked RBMs or autoencoders [12, 36] or DBMs [30], and more recently, this has taken the form of generative adversarial models [9]. This line of work, however, has focused on building a 2D generative model of the pixels themselves. In this case, if the representation captures any 3D properties, it is modeled implicitly. In contrast, we focus on explicitly modeling the 3D shape of the world. Thus, our work is most similar to a number of recent exceptions to the 2D end-to-end approach. Dosovitskiy *et al.* [5] used 3D CAD models to learn a parameterized generative model for objects and Kulkarni *et al.* [19] introduced a technique to guide the latent representation of a generative model to explicitly model certain 3D properties. While they use 3D data like our work, they use it to build a generative model for 2D images. Our work is complementary: their work can generate the pixels for a chair and ours can generate the voxels (and thus, help an agent or robot to interact with it).

There has been comparatively less work in the 3D generative space. Past works have used part-based models [2, 16] and deep networks [39, 20, 24] for representing 3D models. In contrast to 2D generative models, these approaches acknowledges the 3D structure of the world. However, unlike our work, it does not address the mapping from images to this 3D structure. We believe this is a crucial distinction: while the world is 3D, the images we receive are intrinsically 2D and we must build our representations with this in mind.

The task of inferring 3D properties from images goes back to the very beginning of vision. Learning-based techniques started gaining traction in the mid-2000s [13, 31] by framing it as a supervised problem of mapping images of scenes to 2.5D maps. Among a large body of works trying to infer 3D representations from images, our approach is most related to a group of works using renderings of 3D CAD models to predict properties such as object viewpoint [35] or class [34], among others [33, 10, 27]. Typically, these approaches focus on global 3D properties such as pose in the case of objects, and 2.5D maps in the case of scenes. Our work predicts a much more challenging representation, a voxel map (i.e., including the occluded parts). Related works in 3D prediction include [17, 38, 3]. Our approach differs from these as it is class agnostic, voxel based and learns a joint embedding that enables various applications beyond 3D prediction.

Our final output is related to CAD model retrieval in the sense that one output of our approach is a 3D model. Many approaches achieve this via alignment [23, 1, 14] or joint, but non-generative embeddings [21]. In contrast to these works, we take the extreme approach of generating the 3D voxel map from the image. While we obtain coarser results than using an existing model, this explict generative mapping gives the potential to generalize to previously unseen objects.

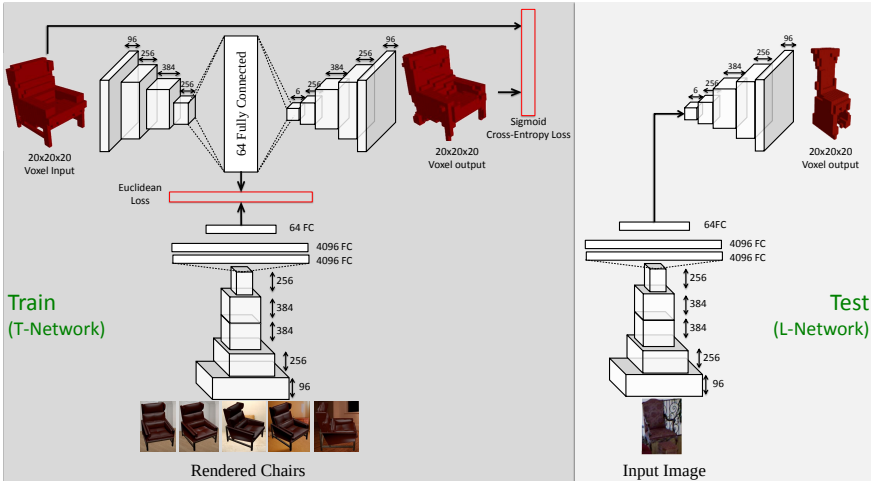


Fig. 2: Our proposed TL-embedding network. **(a) T-network:** At training time, the network takes two inputs: 2D RGB images which are fed into ConvNet at the bottom and 3D voxel maps which are fed into the autoencoder on the left. The output is a 3D voxel map. We apply two losses jointly: a reconstruction loss for the voxel outputs, and a regression loss for the 64-D embedding in the middle. **(b) L-network:** During testing, we remove the encoder part and only use the image as input. The ConvNet predicts the embedding representation and the decoder predicts the voxel.

3 Our Approach

To reiterate, our goal is to learn a vector representation that is: (a) **generative:** we should be able to generate voxels in 3D from this representation; and (b) **predictable:** we should be able to take a 2D image of an object and predict this representation. Both properties are vital for image understanding tasks.

We propose a novel TL-embedding network (Fig. 2) to optimize both these criteria. The T and L refer to the architecture in the training and testing phase. The top part of the T network is an autoencoder with convolution and deconvolution layers. The encoder maps the 3D voxel map to a low-dimensional subspace. The decoder maps a datapoint in the low-dimensional subspace to a 3D voxel map. The autoencoder forces the embedding to be generative, and we can sample datapoints in this embedding to reconstruct new objects. To optimize the predictable criterion, we use a ConvNet architecture similar to AlexNet [18], adding a loss function that ensures the embedding space is predictable from pixels.

Training this TL-embedding network requires 2D RGB images and their corresponding 3D voxel maps. Since this data is hard to obtain, we use CAD model datasets to obtain voxel maps and render these CAD models with different random backgrounds to generate corresponding image data. We now describe our network architecture and the details of our training and testing procedure.

Fig. 3: Sample renderings used to train our network. We render each training model into 72 views over a random background each epoch of training.

Autoencoder Network Architecture: The autoencoder takes a $20 \times 20 \times 20$ voxel grid representation of the CAD model as input. The encoder consists of four convolutional layers followed by a fully connected layer that produces an embedding vector. The decoder takes this embedding and maps it to a 20^3 voxel grid with five deconvolutional layers. Throughout, we use 3D convolutions with stride 1, connected via parameterized ReLU [11] non-linearities.

We train the autoencoder with a Cross-Entropy loss on the final voxel output against the original voxel input. This loss function has the form:

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

where p_n is the target probability (1 or 0) of a voxel being filled, $\hat{p}_n$ is the predicted probability obtained through a sigmoid, and $N = 20^3$.

Mapping 2D Image to Embedding Space: The lower part of the T network learns a mapping from 2D image space to the 64D embedding space. We adopt the AlexNet architecture [18] which has five convolutional layers and two fully connected layers. We add a 64D fc8 layer to the original AlexNet architecture and use a Euclidean loss. We initialize this network with the parameters trained on ImageNet [4] classification task.

One strength of our TL-embedding network is that it can be used to predict a 3D voxel map for a given 2D image. At test time, we remove the encoder part of the autoencoder network and connect the output of the image embedding network to the decoder to obtain this voxel output.

3.1 Training the TL-Embedding Network

We train the network using batches of (image, voxel) pairs. The images are generated by rendering the 3D model and the network is then trained in a three stage procedure. We now describe this in detail.

Data Generation: We use ideas from [35] to render the 3D models for training our network. To prevent the network from overfitting to sharp edges when rendered on a plain background, we render it on randomly selected open room images downloaded from the internet. Following the popular practice [34], we render all the models into 72 views, at three elevations of 15, 30 and 45 degrees and 24 azimuth angles from 0 to 360 degrees, in increments of 15 degrees. We convert the 3D models into 20^3 voxel grid using the voxelizer from [39].

Table 1: Reconstruction performance using AP on test data.

	Chair	Table	Sofa	Cabinet	Bed	Overall
Proposed (before Joint)	**96.4**	**97.1**	99.1	**99.3**	**94.1**	**97.6**
Proposed (after Joint)	**96.4**	97.0	**99.2**	**99.3**	93.8	**97.6**
PCA	94.8	96.7	98.6	99.0	91.5	96.8

Three-stage Training: Training a TL-embedding network from scratch and jointly is a challenging problem. Therefore, we take a three stage procedure. (1) In the first stage, we train the autoencoder part of the network independently. This network is initialized at random, and trained end-to-end with the sigmoid cross-entropy loss. We train this for about 200 epochs. (2) In the second stage we train the ConvNet to regress to the 64D representation. Specifically, the encoder generates the embedding for the voxel and the image network is trained to regress the embedding. The image network is initialized using ImageNet pre-trained weights. We keep the lower convolutional layers fixed. (3) In the final stage, we finetune the network jointly with both the losses. In this stage, we observe that the prediction loss reduces significantly while reconstruction loss reduces marginally. We also observe that most of the parameter update happens in the autoencoder network, indicating that the autoencoder updates its latent representation to make it easily predictable from images, while maintaining or improving the reconstruction performance given this new latent representation.
Implementation Details: We implement this network using the Caffe [15] toolbox. In the first stage, we initialize all layers of autoencoder network from scratch using $\mathcal{N}(0, 0.01)$ and train with a uniform learning rate of 10^{-6}. Next, we train the image network by initializing fc8 from scratch and remaining layers from ImageNet. We finetune all layers after and including conv4 with a uniform learning rate of 10^{-8}. A lower learning rate is required because the initial prediction loss values are in the range of 500K. The encoder network from the autoencoder is used in testing-phase with its previously learned weights to generate the labels for image network. Finally, we jointly train using both losses, initializing the network using weights learned earlier, and finetuning all layers of autoencoder and all layers after and including conv4 for image network with a learning rate of 10^{-10}. Since our network now has two losses, we balance their values by scaling the autoencoder loss to have approximately same initial value, as otherwise the network tends to optimize for the prediction loss without regard to the reconstruction loss.

4 Experiments

We now experimentally evaluate the method. Our overarching goal is to answer the following questions: (1) is the representation we learn generative in 3D? (2) can the representation be predicted from images in 2D? In addition to directly answering these questions, we verify that the model has learned a sensible latent

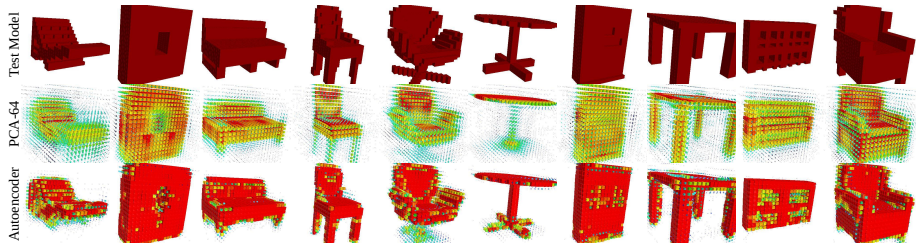


Fig. 4: Reconstructions of *random* test models using PCA and the autoencoder. Predicted voxels are colored and sized by confidence of prediction, from large and red to small and blue in decreasing order of confidence. PCA is much less confident about the extent as well as fine details as compared to our autoencoder.

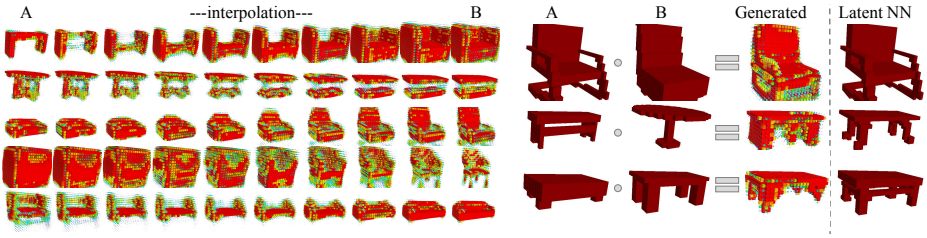
representation by ensuring that the latent representation satisfies a number of properties, such as being smooth, discriminative and allowing arithmetic.

We note that our approach has a capability that, to the best of our knowledge, is previous unexplored: it can simultaneously reconstruct in 3D and predict from 2D. Thus, there are no standard baselines or datasets for this task. Instead, we adopt standard datasets for each of the many tasks that our model can perform. Where appropriate, we compare the method with existing methods. These baselines, however, are specialized solutions to only one of the many tasks we can solve and often use additional supervisory information. As the community starts tackling increasingly difficult 3D problems like direct voxel prediction, we believe that our work can be a strong baseline to benchmark progress.

We proceed as follows. We introduce the datasets and evaluation criterion that we use in Sec. 4.1. We first verify that our learned representation models the space of voxels well in a number of ways: that it is reconstructive, smooth, and can be used to distinguish different classes of objects (Sec. 4.2). This evaluates the representation independently of its ability to predict voxels from images. We then verify that our approach can predict the voxels from 2D and show that it outperforms alternate options (Sec. 4.3). Subsequently, we show that our representation can be used to do CAD retrieval from natural images (Sec. 4.4) and is capable of performing 3D shape arithmetic (Sec. 4.5).

4.1 Datasets and Evaluation

We use two datasets for evaluation. The first is a CAD model dataset used to train the TL-embedding and to explore the learned embedding. The second is an in-the-wild dataset used to verify that the approach works on natural images.
CAD Dataset: We use CAD models from the ShapeNet[39] database. This database contains over 220K models organized into 3K WordNet synsets. We take a set of common indoor objects: chair (6778 models), table (8509 models), sofa (3173 models), cabinet (1572 models), and bed (254 models). We split these models randomly into 16228 train and 4058 test objects. All our models are



(a) (b)

Fig. 5: (a) Reconstructions for linear interpolation between two randomly picked latent representations. (b) Evaluating generative ability by combining dimensions from two training models. We show the reconstruction and the nearest neighbor in the training set (over latent features). The difference shows we can generate novel models, such as an armchair with one arm-rest.

trained with rendered images and voxels from the above train set. We use the test set to quantify our performance and analyze our models.

IKEA Dataset: We quantify the performance of our model on natural indoor images from IKEA Dataset [23] which are labeled with 3D models. Since our approach expects to reconstruct a single object, we test it on cropped images of these objects. These boxes, however, include cluttered backgrounds and pieces of other objects. After cropping these objects out of provided 759 images, we get 937 images labeled with one of provided 225 3D models.

Evaluation Metric: Throughout the paper, we use Average Precision (AP) over the complete test set to evaluate reconstruction performance. We also show per-class APs where applicable to better characterize our model's performance.

4.2 Embedding Analysis

We start by probing our learned representation in terms of 3D voxels. Here, we focus on the autoencoder part of the network – that is, we feed a voxel grid to the network and verify a number of properties: (a) that it can reconstruct the voxels well qualitatively and quantitatively, which verifies that the method works; (b) that it outperforms a linear baseline, PCA, for reconstruction, which further validates the choice of a convolutional autoencoder; and (c) that the learned representation is smooth and carries class-discriminative information, which acts as additional confirmation that the representation is meaningful.

Qualitative Results: First, we show qualitative results: Fig. 4 shows randomly selected reconstructions using the autoencoder and PCA. While a simple linear approach is sufficient to capture the coarse structure, our approach does much better at fine-details (chair legs in col. 6) as well as at getting the extent correct (back of the chair in col. 4). Note also the large amount of low but non-zero probability voxels in the free-space in PCA compared to the auto-encoder.

We next show that the learned space is smooth, by computing reconstructions for linear interpolation between latent representations of randomly picked test models. As Fig. 5(a) shows, the 3D models smoothly transition in structure and most intermediate models are also physically plausible. We also show results exploring the learned space and verifying whether the dimensions are meaningful. One way to do this is to generate new points in the space and reconstruct them. We generate these points by taking the first 32 dimensions from one model and the rest from another. As seen by the difference between the reconstruction and the nearest model in Fig. 5(b), this can generate previously unseen models that combine aspects of each model.

We further attempt to understand the embedding space by clamping all the dimensions of a latent vector but one and scaling the selected dimension by adding a fixed value to it. We show its effect on two dimensions and three models in Fig. 6. Such scaling of these dimensions produces consistent effects across models, suggesting that some learned dimensions are semantically meaningful.

Quantitative Reconstruction Accuracy: We now evaluate the reconstruction performance quantitatively on the CAD test data and report results in Table 1. Our goal here is to verify that the auto-encoder is worthwhile: we thus compare to PCA using the same number of dimensions. Our method obtains extremely high performance, 97.6% AP and consistently outperforms PCA, reducing the average error rate by 25% relative. It can be seen in Table 1 that some categories are easier than others: sofas and cabinets are naturally more easy than beds (including bunk-beds) and chairs. Our method consistently obtains larger gains on challenging objects, indicating the merits of a non-linear representation. We also evaluate the performance of the autoencoder after the joint training. Even after being optimized to be more predictable from image space, we can see that it still preserves the overall reconstruction performance.

CAD Classification: If our representation models 3D well, it should permit us to distinguish different types of objects. We empirically verify this by using our approach *without modifications* as a representation to classify 3D shapes. Note that while adding a classification loss and finetuning might further improve results, it would defeat the purpose of this experiment, which is to see whether the model learns a good 3D representation on its own. We evaluate our representation's performance for a classification task on the Princeton ModelNet40 [28] dataset with standard train-test split from [39]. We train the network on all 40 classes (again: no class information is provided) and then use the autoencoder representation as a feature for 40-way classification. Since our representation is low-dimensional (64D), we expand the feature to include pairwise features and train a linear SVM. Our approach obtains an accuracy of 74.4%. This is within 2.6% of [39], a recent approach on voxels that uses class information at representation-learning time, and finetunes the representation discriminatively for the classification experiment. Using a 64D PCA representation trained on ModelNet40 trainset with the same feature augmentation and linear SVM obtains 68.4%. This shows that our representation is class-discriminative despite not being trained or designed so, and outperforms the PCA.

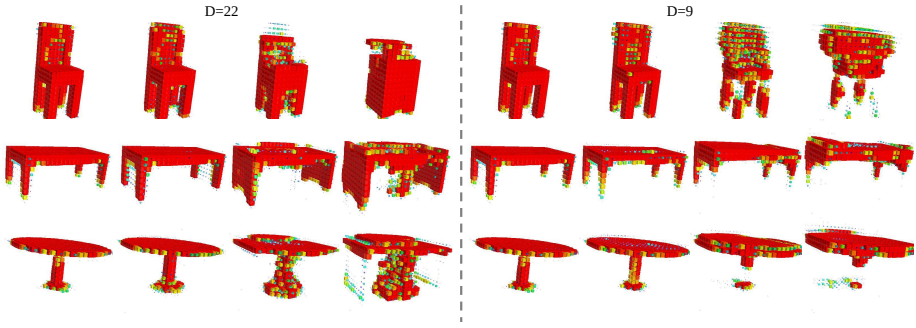


Fig. 6: We evaluate if the dimensions are meaningful by scaling each dimension separately and analyzing the effect on the reconstruction. Some dimensions have a consistent effect on reconstruction across objects. Higher values in dimension 22 lead to thicker legs, and higher values in 9 lead to disappearance of legs.

Table 2: Average Precision for Voxel Prediction on the CAD test set. The Proposed TL-Network outperforms the baselines on each object.

	Chair	Table	Sofa	Cabinet	Bed	Average
Proposed (with Joint)	**66.9**	**59.7**	**79.3**	**79.3**	**41.9**	**65.4**
Proposed (without Joint)	66.6	57.5	**79.3**	76.5	33.8	62.7
Direct-conv4	40.9	23.7	58.1	44.3	23.1	38.0
Direct-fc8	21.8	15.5	35.6	32.7	18.6	24.8

4.3 Voxel Prediction

We now turn to the task of predicting a 3D voxel grid from an image. We obtain strong performance on this task and outperform a number of baselines, demonstrating the importance of each part of our approach.

Baselines: To the best of our knowledge, there are no methods that directly predict voxels from an image; we therefore compare to a direct prediction method as well an ablation study, where we do not perform joint training. Specifically: (a) *Direct*: finetuning the ImageNet pre-trained AlexNet to predict the 20^3 voxel grid directly. This corresponds to removing the auto-encoder. We tried two strategies for freezing the layers: Direct-conv4 refers to freezing all layers before conv4 and Direct-fc8 refers to freezing all layers except fc8. (b) *Without Joint*: training the T-L network without the final joint fine-tuning (i.e., following only the first two training stages). The direct baselines test whether the auto-encoder's low-dimensional representation is necessary and the without-joint tests whether learning the model to be jointly generative *and* predictable is important.

Qualitative Results: We first show qualitative results on natural images in Fig. 7. Note that our method automatically predicts occluded regions of the object, unlike most work on single image 3D (e.g., [31, 13, 7, 6, 8, 37]) that predict a 2.5D shell. For instance, our method predicts all four legs of furniture even if fewer are

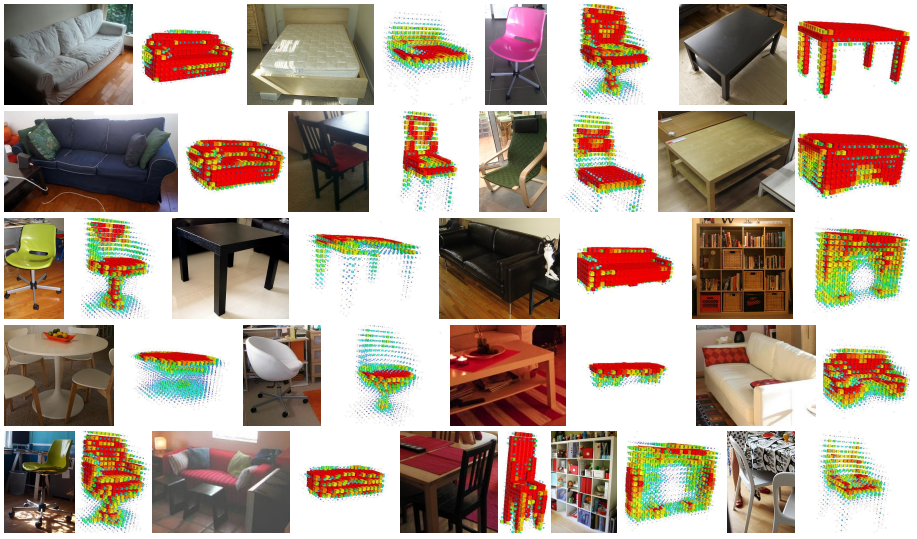


Fig. 7: Reconstruction results on the IKEA dataset. Our model generalizes well to real images, even to bookshelves which our model is not trained on.

Table 3: Average Precision for Voxel Prediction on the IKEA dataset.

	Bed	Bookcase	Chair	Desk	Sofa	Table	Overall
Proposed	**56.3**	**30.2**	**32.9**	25.8	**71.7**	**23.3**	**38.3**
Direct-conv4	38.2	26.6	31.4	**26.6**	69.3	19.1	31.1
Direct-fc8	29.5	17.3	20.4	19.7	38.8	16.0	19.8

visible. Our model generalizes well to natural images even though it was trained on CAD models. Note that for instance, the round and rectangular tables are predicted as being round and rectangular, and office chairs on a single post and four-legged chairs can be distinguished. One difficulty with this data is that objects are truncated or occluded and some windows contain multiple objects; our model does well on this data, nonetheless.

Quantitative Results: We now evaluate the approach quantitatively on both datasets. We report results on the CAD dataset in Table 2. Our approach outperforms all the baselines. Directly predicting the voxels does substantially worse because predicting all the voxels is a very difficult task compared to our embedding space. Not doing joint training produces worse results because the embedding is not forced to be predictable.

The IKEA dataset is more challenging because it is captured in-the-wild, but our approach still produces quantitatively strong performance. While the CAD Dataset models are represented in canonical form, the IKEA models are provided in no consistent orientation. We thus attempt to align each prediction with the

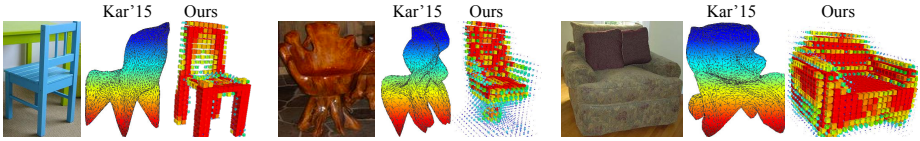


Fig. 8: Predictions on PASCAL 3D+ images using [17] and our method. Our method is better at capturing fine stylistic details, like the straight legs and the hollow back in the first case, a single central leg in the second, and no visible legs in the last.

Fig. 9: Top CAD model retrievals from natural images from the IKEA dataset.

ground-truth model by taking the best rigid alignment over permutations, flips and translational alignments (up to 10%) of the prediction. As Table 3 shows, our approach outperforms the direct prediction by a large margin (38% compared to 31%). If we do not correct for translational alignments, we still outperform the baseline (33% vs 28%). Directly predicting voxels again performs worse compared to predicting the latent space and reconstructing, validating the idea of using a lower-dimensional representation of objects.

Comparison with Kar *et al.* [17]: We also compare our method with [17] on PASCAL 3D+ v1.0 [40] dataset for categories that overlap with our training categories (chair and sofa). As Fig. 8 shows, our output is more varied and captures stylistic details better. For quantitative comparison, we voxelize their output and ground truth, and compute the overlap P-R curve with alignment. Since [17] produces a binary non-probabilistic prediction and thus yields only one operating point, we compare via maximum F-1 score instead of AP. After aligning, we outperform their method 0.492 to 0.463.

4.4 CAD Retrieval

We now show results for retrieving CAD models from natural images. Our system can naturally tackle this task: we map each model in the CAD corpus as well as

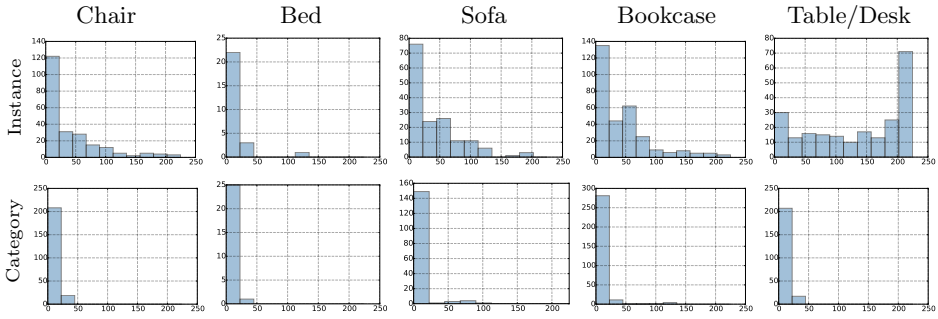


Fig. 10: Histograms over position in retrieval list obtained by our proposed approach (Y axis: #images, X axis: position). First row of histograms is over the position of instance match, and second is over position of category match.

Table 4: Mean recall @10 of ground truth model in retrievals for our method and baseline described in Sec. 4.4

	Sofa	Chair	Bookcase	Bed	Table	Overall
Proposed	**32.3**	**41.0**	**26.8**	**38.5**	8.0	**29.3**
Fc7-NN	14.6	33.9	23.5	7.7	**17.4**	19.4

the image to their latent representations, and perform a nearest neighbor search in this embedding space.

We use cosine distance in the latent space for retrieval. This approach is complementary to approaches like [23, 22]: these approaches assume the existence of an exact-match 3D model and fits the 3D model into the image. Our approach, on the other hand, does not assume exact match and thus generalizes to retrieving the most similar object to the depicted object (i.e., what is the next-most similar object in the corpus). We show qualitative results in Fig. 9.

We now quantitatively evaluate our approach. For each test window, we rank all 225 CAD models in the corpus by cosine distance. We can then determine two quantities: *(a) Instance match:* at what rank does the exact-match CAD model appear? *(b) Category match:* at what rank does the first model of the same category appear? As a baseline, we render all the 225 models at 30 deg. elevation and 8 uniformly sampled azimuths from 0 to 360 deg. onto a white background, after scaling and translating each model to a unit square at the origin. We then use ImageNet trained AlexNet's fc7 features over the query image and renderings to perform nearest neighbor search (cosine distance). The first position at which a rendering of a model appears in the retrievals is taken as the position for that model. Note that this is a strong baseline with access to lot more information since it sees images, which are much higher resolution than our 20^3 voxel grids. Moreover, it is significantly slower than our method, as it represents each 3D model using 8 vectors of 4096D each, while our approach uses only a single 64D

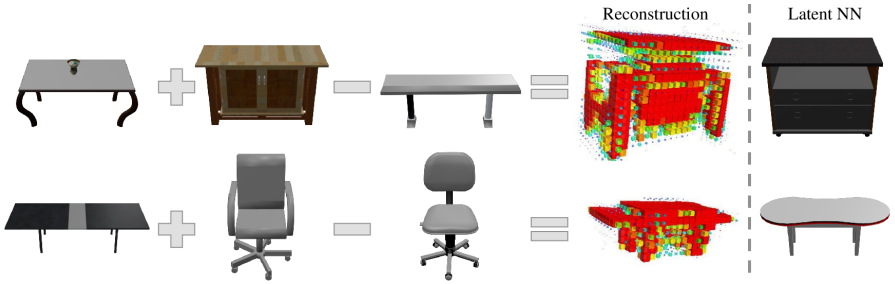


Fig. 11: Results of shape arithmetic. In the first case, adding a cabinet-like-table to a table and removing small 2-leg table results in a table with built-in cabinet. In the second case, adding and removing a similar looking chair with straight and curved edges respectively leads to a table with curved edges.

vector. As shown in Table 4, which reports the mean recall@10 of instance match, we outperform this baseline on all categories except tables/desks because most of the table models are very similar, and fine differentiation between specific models is very hard for a coarse 20^3 voxel representation. We report histograms of these ranks in Fig. 10 per object category. For many categories, the top response is the correct category, and the exact-match model is typically ranked highly. Poor performance tends to result from images containing multiple objects (e.g., a table picture with chairs in it), causing the network to predict the representation for the "wrong" object out of the ambiguous input. We also compare our model with [21] in the supplement available on the project webpage.

4.5 Shape Arithmetic

We have shown that the latent space is reconstructive and smooth, that it is predictable, and that it carries class information. We now show some attempts at probing the learned representation. Previous work in vector embedding spaces [29, 25] exhibit the phenomena of being able to perform arithmetic on these vector representations. For example, [25] showed that vector(King) - vector(Man) + vector(Woman) results in vector whose nearest neighbor was the vector for Queen. We perform a similar experiment by randomly selecting triplets of 3D models and performing this $a + b - c$ operation on their latent representations. We then use the resulting feature to generate the voxel representation and also find the nearest neighbor in the dataset over cosine distance on this latent representation. We show some interesting triplets in Fig. 11.

Acknowledgments: This work was partially supported by Siebel Scholarship to RG, NDSEG Fellowship to DF and Bosch Young Faculty Fellowship to AG. This material is based on research partially sponsored by ONR MURI N000141010934, ONR MURI N000141612007, NSF1320083 and a gift from Google. The authors would like to thank Yahoo! and Nvidia for the compute cluster and GPU donations respectively. The authors would also like to thank Martial Hebert and Xiaolong Wang for many helpful discussions.

References

1. Aubry, M., Maturana, D., Efros, A., Russell, B., Sivic, J.: Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of cad models. In: CVPR (2014)
2. Chaudhuri, S., Kalogerakis, E., Guibas, L., Koltun, V.: Probabilistic reasoning for assembly-based 3D modeling. SIGGRAPH (2011)
3. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. CoRR abs/1604.00449 (2016)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
5. Dosovitskiy, A., Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: CVPR (2015)
6. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV (2015)
7. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS (2014)
8. Fouhey, D.F., Gupta, A., Hebert, M.: Data-driven 3D primitives for single image understanding. In: ICCV (2013)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
10. Gupta, S., Arbeláez, P.A., Girshick, R.B., Malik, J.: Inferring 3D object pose in RGB-D images. CoRR (2015)
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR abs/1502.01852 (2015)
12. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science (2006)
13. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. In: IJCV (2007)
14. Huang, Q., Wang, H., Koltun, V.: Single-view reconstruction via joint analysis of image and shape collections. SIGGRAPH 34(4) (2015)
15. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
16. Kalogerakis, E., Chaudhuri, S., Koller, D., Koltun, V.: A Probabilistic Model of Component-Based Shape Synthesis. SIGGRAPH (2012)
17. Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-specific object reconstruction from a single image. In: CVPR (2015)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012)
19. Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. In: NIPS (2015)
20. Li, Y., Pirk, S., Su, H., Qi, C.R., J., G.L.: FPNN: Field probing neural networks for 3d data. CoRR abs/1605.06240 (2016)
21. Li, Y., Su, H., Qi, C.R., Fish, N., Cohen-Or, D., Guibas, L.J.: Joint embeddings of shapes and images via cnn image purification. ACM TOG (2015)
22. Lim, J.J., Khosla, A., Torralba, A.: FPM: fine pose parts-based model with 3D CAD models. In: ECCV (2014)
23. Lim, J.J., Pirsiavash, H., Torralba, A.: Parsing IKEA objects: Fine pose estimation. In: ICCV (2013)

24. Maturana, D., Scherer, S.: VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In: IROS (2015)
25. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
26. Olshausen, B., Field, D.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature (1996)
27. Peng, X., Sun, B., Ali, K., Saenko, K.: Exploring invariances in deep convolutional neural networks using synthetic images. CoRR (2014)
28. Princeton ModelNet: http://modelnet.cs.princeton.edu/
29. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR abs/1511.06434 (2015)
30. Salakhutdinov, R., Hinton, G.: Deep Boltzmann machines. In: AISTATS. vol. 5 (2009)
31. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D scene structure from a single still image. TPAMI 30(5), 824–840 (2008)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
33. Stark, M., Goesele, M., Schiele, B.: Back to the future: Learning shape models from 3D CAD data. In: BMVC (2010)
34. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV (2015)
35. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In: ICCV (2015)
36. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. (2010)
37. Wang, X., Fouhey, D.F., Gupta, A.: Designing deep networks for surface normal estimation. In: CVPR (2015)
38. Wu, J., Xue, T., Lim, J.J., Tian, Y., Tenenbaum, J.B., Torralba, A., Freeman, W.T.: Single image 3d interpreter network. In: ECCV (2016)
39. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: A deep representation for volumetric shapes. In: CVPR (2015)
40. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: A benchmark for 3d object detection in the wild. In: WACV (2014)