Learning Representations and Generative Models for 3D Point Clouds

Panos Achlioptas [1] **Olga Diamanti** [1] **Ioannis Mitliagkas** [2] **Leonidas Guibas** [1]

Abstract

Three-dimensional geometric data offer an excellent domain for studying representation learning and generative modeling. In this paper, we look at geometric data represented as point clouds. We introduce a deep AutoEncoder (AE) network with state-of-the-art reconstruction quality and generalization ability. The learned representations outperform existing methods on 3D recognition tasks and enable shape editing via simple algebraic manipulations, such as semantic part editing, shape analogies and shape interpolation, as well as shape completion. We perform a thorough study of different generative models including GANs operating on the raw point clouds, significantly improved GANs trained in the fixed latent space of our AEs, and Gaussian Mixture Models (GMMs). To quantitatively evaluate generative models we introduce measures of sample fidelity and diversity based on matchings between sets of point clouds. Interestingly, our evaluation of generalization, fidelity and diversity reveals that GMMs trained in the latent space of our AEs yield the best results overall.

1. Introduction

Three-dimensional (3D) representations of real-life objects are a core tool for vision, robotics, medicine, augmented reality and virtual reality applications. Recent attempts to encode 3D geometry for use in deep learning include view-based projections, volumetric grids and graphs. In this work, we focus on the representation of 3D point clouds. Point clouds are becoming increasingly popular as a homogeneous, expressive and compact representation of surface-based geometry, with the ability to represent geometric details while taking up little space. Point clouds are particularly amenable to simple geometric operations and are a standard 3D acquisition format used by range-scanning devices like LiDARs, the Kinect or iPhone's face ID feature.

[1]Department of Computer Science, Stanford University, USA [2]MILA, Department of Computer Science and Operations Research, University of Montréal, Canada. Correspondence to: Panos Achlioptas <optas@cs.stanford.edu>.

All the aforementioned encodings, while effective in their target tasks (e.g. rendering or acquisition), are hard to manipulate directly in their raw form. For example, naïvely interpolating between two cars in any of those representations does not yield a representation of an "intermediate" car. Furthermore, these representations are not well suited for the design of generative models via classical statistical methods. Using them to edit and design new objects involves the construction and manipulation of custom, object-specific parametric models, that link the semantics to the representation. This process requires significant expertise and effort.

Deep learning brings the promise of a *data-driven approach*. In domains where data is plentiful, deep learning tools have eliminated the need for hand-crafting features and models. Architectures like AutoEncoders (AEs) (Rumelhart et al., 1988; Kingma & Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Radford et al.; Che et al., 2016) are successful at learning data representations and generating realistic samples from complex underlying distributions. However, an issue with GAN-based generative pipelines is that training them is notoriously hard and unstable (Salimans et al., 2016). In addition, and perhaps more importantly, *there is no universally accepted method for the evaluation of generative models*.

In this paper, we explore the use of deep architectures for *learning representations* and introduce the first deep *generative models* for *point clouds*. Only a handful of deep architectures tailored to 3D point clouds exist in the literature, and their focus is elsewhere: they either aim at classification and segmentation (Qi et al., 2016a; 2017), or use point clouds *only* as an intermediate or output representation (Kalogerakis et al., 2016; Fan et al., 2016). Our specific contributions are:

- A new AE architecture for point clouds—inspired by recent architectures used for classification (Qi et al., 2016a)—that can learn compact representations with (i) good reconstruction quality on unseen samples; (ii) good classification quality via simple methods (SVM), outperforming the state of the art (Wu et al., 2016); (iii) the capacity for meaningful semantic operations, interpolations and shape-completion.

- The first set of deep generative models for point clouds, able to synthesize point clouds with (i) measurably

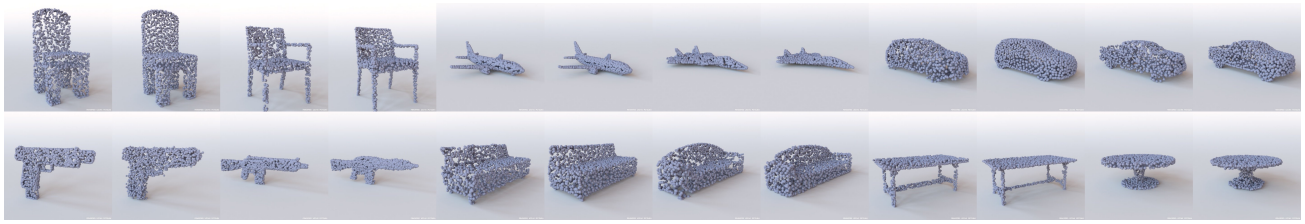


Figure 1. Reconstructions of unseen shapes from the test split of the input data. The leftmost image of each pair shows the ground truth shape, the rightmost the shape produced after encoding and decoding using a *class-specific* AE-EMD.

high fidelity to, and (ii) good coverage of both the training and the held-out data. One workflow that we propose is to first train an AE to learn a latent representation and then train a generative model in that fixed latent space. The GANs trained in the latent space, dubbed here *l-GANs*, are easier to train than raw GANs and achieve superior reconstruction and better coverage of the data distribution. Multi-class GANs perform almost on par with class-specific GANs when trained in the latent space.

- A study of various old and new point cloud metrics, in terms of their applicability (i) as reconstruction objectives for learning good representations; (ii) for the evaluation of generated samples. We find that a commonly used metric, Chamfer distance, fails to identify certain pathological cases.

- Fidelity and coverage metrics for generative models, based on an optimal matching between two different collections of point clouds. Our coverage metric can identify parts of the data distribution that are completely missed by the generative model, something that diversity metrics based on cardinality might fail to capture (Arora & Zhang, 2017).

The rest of this paper is organized as follows: Section 2 outlines some background for the basic building blocks of our work. Section 3 introduces our metrics for the evaluation of generative point cloud pipelines. Section 4 discusses our architectures for latent representation learning and generation. In Section 5, we perform comprehensive experiments evaluating all of our models both quantitatively and qualitatively. Further results can be found in the Appendix. Last, the code for all our models is publicly available[1].

2. Background

In this section we give the necessary background on point clouds, their metrics and the fundamental building blocks that we will use in the rest of the paper.

[1] http://github.com/optas/latent_3d_points

2.1. Point clouds

Definition A point cloud represents a geometric shape—typically its surface—as a set of 3D locations in a Euclidean coordinate frame. In 3D, these locations are defined by their x, y, z coordinates. Thus, the point cloud representation of an object or scene is a $N \times 3$ matrix, where N is the number of points, referred to as the point cloud resolution.

Point clouds as an input modality present a unique set of challenges when building a network architecture. As an example, the convolution operator—now ubiquitous in image-processing pipelines—requires the input signal to be defined on top of an underlying grid-like structure. Such a structure is not available in raw point clouds, which renders them significantly more difficult to encode than images or voxel grids. Recent classification work on point clouds (PointNet (Qi et al., 2016a)) bypasses this issue by avoiding convolutions involving groups of points. Another related issue with point clouds as a representation is that they are permutation invariant: any reordering of the rows of the point cloud matrix yields a point cloud that represents the same shape. This property complicates comparisons between two point sets which is needed to define a reconstruction loss. It also creates the need for making the encoded feature permutation invariant.

Metrics Two permutation-invariant metrics for comparing unordered point sets have been proposed in the literature (Fan et al., 2016). On the one hand, the *Earth Mover's* distance (EMD) (Rubner et al., 2000) is the solution of a transportation problem which attempts to transform one set to the other. For two equally sized subsets $S_1 \subseteq R^3, S_2 \subseteq R^3$, their EMD is defined by

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

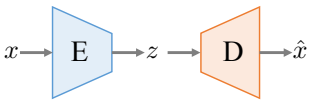
where ϕ is a bijection. As a loss, EMD is differentiable almost everywhere. On the other hand, the *Chamfer* (pseudo)-distance (CD) measures the squared distance between each point in one set to its nearest neighbor in the other set:

$$d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2.$$

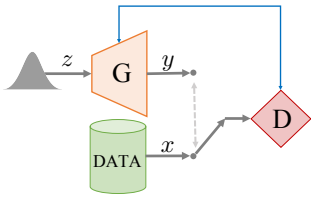
CD is differentiable and compared to EMD more efficient to compute.

2.2. Fundamental building blocks

Autoencoders One of the main deep-learning components we use in this paper is the *AutoEncoder* (AE, inset), which is an architecture that learns to reproduce its input. AEs can be especially useful, when they contain a narrow *bottleneck layer* between input and output. Upon successful training, this layer provides a low-dimensional representation, or *code*, for each data point. The Encoder (E) learns to compress a data point $\boldsymbol{x}$ into its latent representation, $\boldsymbol{z}$. The Decoder (D) can then produce a reconstruction $\hat{\boldsymbol{x}}$, of $\boldsymbol{x}$, from its encoded version $\boldsymbol{z}$.



Generative Adversarial Networks In this paper we also work with Generative Adversarial Networks (GANs), which are state-of-the-art generative models. The basic architecture (inset) is based on a adversarial game between a *generator* (G) and a *discriminator* (D). The generator aims to synthesize samples that look indistinguishable from real data (drawn from $\boldsymbol{x} \sim p_{\text{data}}$) by passing a randomly drawn sample from a simple distribution $\boldsymbol{z} \sim p_z$ through the generator function. The discriminator is tasked with distinguishing synthesized from real samples.



Gaussian Mixture Model A GMM is a probabilistic model for representing a population whose distribution is assumed to be multimodal Gaussian, i.e. comprising of multiple subpopulations, where each subpopulation follows a Gaussian distribution. Assuming the number of subpopulations is known, the GMM parameters (means and variances of the Gaussians) can be learned from random samples, using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). Once fitted, the GMM can be used to sample novel synthetic samples.

3. Evaluation Metrics for Generative Models

An important component of this work is the introduction of measures that enable comparisons between two sets of points clouds A and B. These metrics are useful for assessing the degree to which point clouds, synthesized or reconstructed, represent the same population as a held-out test set. Our three measures are described below.

JSD The Jensen-Shannon Divergence between marginal distributions defined in the Euclidean 3D space. Assuming point cloud data that are axis-aligned and a canonical voxel grid in the ambient space; one can measure the degree to which point clouds of A tend to occupy similar locations as those of B. To that end, we count the number of points lying within each voxel across *all* point clouds of A, and correspondingly for B and report the JSD between the obtained empirical distributions (P_A, P_B):

$$JSD(P_A \parallel P_B) = \frac{1}{2}D(P_A \parallel M) + \frac{1}{2}D(P_B \parallel M)$$

where $M = \frac{1}{2}(P_A + P_B)$ and $D(\cdot \parallel \cdot)$ the KL-divergence between the two distributions (Kullback & Leibler, 1951).

Coverage For each point cloud in A we first find its closest neighbor in B. Coverage is measured as the *fraction* of the point clouds in B that were matched to point clouds in A. Closeness can be computed using either the CD or EMD point-set distance of Section 2, thus yielding two different metrics, COV-CD and COV-EMD. A high coverage score indicates that most of B is roughly represented within A.

Minimum Matching Distance (MMD) Coverage does not indicate exactly *how well* the covered examples (point-clouds) are represented in set A; matched examples need not be close. We need a way to measure the *fidelity* of A with respect to B. To this end, we match every point cloud of B to the one in A with the minimum distance (MMD) and report the average of distances in the matching. Either point-set distance can be used, yielding MMD-CD and MMD-EMD. Since MMD relies directly on the distances of the matching, it correlates well with how faithful (with respect to B) elements of A are.

Discussion The complementary nature of MMD and Coverage directly follows from their definitions. The set of point clouds A captures all modes of B with good fidelity when MMD is small *and* Coverage is large. JSD is fundamentally different. First, it evaluates the similarity between A and B in coarser way, via marginal statistics. Second and contrary to the other two metrics, it requires pre-aligned data, but is also computationally friendlier. We have found and show experimentally that it correlates well with the MMD, which makes it an efficient alternative for e.g. model-selection, where one needs to perform multiple comparisons between sets of point clouds.

4. Models for Representation and Generation

In this section we describe the architectures of our neural networks starting from an autoencoder. Next, we introduce a GAN that works directly with 3D point cloud data, as well as a decoupled approach which first trains an AE and

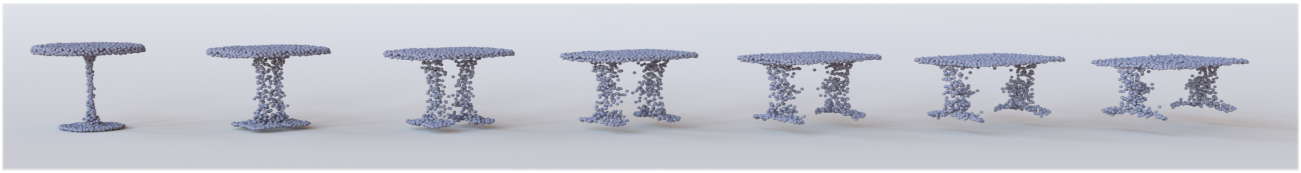


Figure 2. Interpolating between different point clouds, using our latent space representation. More examples for furniture and *human-form* objects (Bogo et al., 2017) are demonstrated in the Appendix in Figures 11 and 14, respectively.

then trains a minimal GAN in the AE's latent space. We conclude with a similar but even simpler solution that relies on classical Gaussian mixtures models.

4.1. Learning representations of 3D point clouds

The input to our AE network is a point cloud with 2048 points (2048×3 matrix), representing a 3D shape. The encoder architecture follows the design principle of (Qi et al., 2016a): 1-D convolutional layers with kernel size 1 and an increasing number of features; this approach encodes every point *independently*. A "symmetric", permutation-invariant function (e.g. a max pool) is placed after the convolutions to produce a joint representation. In our implementation we use 5 1-D convolutional layers, each followed by a ReLU (Nair & Hinton, 2010) and a batch-normalization layer (Ioffe & Szegedy, 2015). The output of the last convolutional layer is passed to a feature-wise maximum to produce a k-dimensional vector which is the basis for our latent space. Our decoder transforms the latent vector using 3 fully connected layers, the first two having ReLUs, to produce a 2048×3 output. For a permutation invariant objective, we explore both the EMD approximation and the CD (Section 2) as our structural losses; this yields two distinct AE models, referred to as AE-EMD and AE-CD. To regularize the AEs we considered various bottleneck sizes, the use of drop-out and on-the-fly augmentations by randomly-rotating the point clouds. The effect of these choices is showcased in the Appendix (Section A) along with the detailed training/architecture parameters. In the remainder of the paper, unless otherwise stated, we use an AE with a 128-dimensional bottleneck layer.

4.2. Generative models for Point Clouds

Raw point cloud GAN (r-GAN) Our first GAN operates on the raw 2048×3 point set input. The architecture of the discriminator is identical to the AE (modulo the filter-sizes and number of neurons), without any batch-norm and with leaky ReLUs (Maas et al., 2013) instead or ReLUs. The output of the last fully connected layer is fed into a sigmoid neuron. The generator takes as input a Gaussian noise vector and maps it to a 2048×3 output via 5 FC-ReLU layers.

Latent-space GAN (l-GAN) For our l-GAN, instead of operating on the raw point cloud input, we pass the data through a pre-trained autoencoder, which is trained separately for each object class with the EMD (or CD) loss function. Both the generator and the discriminator of the l-GAN then operate on the bottleneck variables of the AE. Once the training of GAN is over, we convert a code learned by the generator into a point cloud by using the AE's decoder. Our chosen architecture for the l-GAN, which was used throughout our experiments, is *significantly* simpler than the one of the r-GAN. Specifically, an MLP generator of a single hidden layer coupled with an MLP discriminator of two hidden layers suffice to produce measurably good and realistic results.

Gaussian mixture model In addition to the l-GANs, we also fit a family of Gaussian Mixture Models (GMMs) on the latent spaces learned by our AEs. We experimented with various numbers of Gaussian components and diagonal or full covariance matrices. The GMMs can be turned into point cloud generators by first sampling the fitted distribution and then using the AE's decoder, similarly to the l-GANs.

5. Experimental Evaluation

In this section we experimentally establish the validity of our proposed evaluation metrics and highlight the merits of the AE-representation (Section 5.1) and the generative models (Section 5.2). In all experiments in the main paper, we use shapes from the ShapeNet repository (Chang et al., 2015), that are axis aligned and centered into the unit sphere. To convert these shapes (meshes) to point clouds we uniformly sample their faces in proportion to their area. Unless otherwise stated, we train models with point clouds from a single object class and work with train/validation/test sets of an 85%-5%-10% split. When reporting JSD measurements we use a 28^3 regular voxel grid to compute the statistics.

5.1. Representational power of the AE

We begin with demonstrating the merits of the proposed AE. First we report its generalization ability as measured using the MMD-CD and MMD-EMD metrics. Next, we utilize its latent codes to do semantically meaningful operations. Finally, we use the latent representation to train SVM classifiers and report the attained classification scores.

Figure 3. Editing parts in point clouds using simple additive algebra on the AE latent space. Left to right: tuning the appearance of cars towards the shape of convertibles, adding armrests to chairs, removing handle from mug. Note that the height of chairs with armrests is on average 13% shorter than of chairs without one; which is reflected also in these results.

Generalization ability. Our AEs are able to reconstruct unseen shapes with quality almost as good as that of the shapes that were used for training. In Fig. 1 we use our AEs to encode unseen samples from the *test* split (the left of each pair of images) and then decode them and compare them visually to the input (the right image). To support our visuals quantitatively, in Table 1 we report the MMD-CD and MMD-EMD between reconstructed point clouds and their corresponding ground-truth in the train and test datasets of the chair object class. The generalization gap under our metrics is small; to give a sense of scale for our reported numbers, note that the MMD is 0.0003 and 0.033 under the CD and EMD respectively between two versions of the test set that only differ by the randomness introduced in the point cloud sampling. Similar conclusions regarding the generalization ability of the AE can be made based on the reconstruction loss attained for each dataset (train or test) which is shown in Fig. 9 of the Appendix.

AE loss	MMD-CD		MMD-EMD	
	Train	Test	Train	Test
CD	**0.0004**	**0.0012**	0.068	0.075
EMD	0.0005	0.0013	**0.042**	**0.052**

Table 1. Generalization of AEs as captured by MMD. Measurements for reconstructions on the training and test splits for an AE trained with either the CD or EMD loss and data of the chair class; Note how the MMD favors the AE that was trained with the same loss as the one used by the MMD to make the matching.

Latent space and linearity. Another argument against under/over-fitting can be made by showing that the learned representation is amenable to intuitive and semantically rich operations. As it is shown in several recent works, well trained neural-nets learn a latent representation where additive linear algebra works to that purpose (Mikolov et al., 2013; Tasse & Dodgson, 2016). First, in Fig. 2 we show linear interpolations, in the latent space, between the left and right-most geometries. Similarly, in Fig. 3 we alter the input geometry (left) by adding, in latent space, the mean vector of geometries with a certain characteristic (e.g., convertible cars or cups without handles). Additional operations (e.g. shape analogies) are also possible, but due to space limitations we illustrate and provide the details in the Appendix (Section B) instead. These results attest to the smoothness of the learned space but also highlight the intrinsic capacity of point clouds to be smoothly morphed.

Shape completions. Our proposed AE architecture can be used to tackle the problem of shape completion with minimal adaptation. Concretely, instead of feeding and reconstructing the same point cloud, we can feed the network with an *incomplete* version of its expected output. Given proper training data, our network learns to complete severely partial point clouds. Due to space limitations we give the exact details of our approach in the Appendix (Section D) and demonstrate some achieved completions in Fig. 4 of the main paper.

Classification. Our final evaluation for the AE's design and efficacy is done by using the learned latent codes as features for classification. For this experiment to be meaningful, we train an AE across all different shape categories: using 57,000 models from 55 categories of man-made objects. Exclusively for this experiment, we use a bottleneck of 512 dimensions and apply random rotations to the input point clouds along the gravity axis. To obtain features for an input 3D shape, we feed its point cloud into the AE and extract the bottleneck activation vector. This vector is then classified by a linear SVM trained on the de-facto 3D classification benchmark of ModelNet (Wu et al., 2015). Table 2 shows comparative results. Remarkably, in the ModelNet10 dataset, which includes classes (chairs, beds etc.) that are populous in ShapeNet, our simple AE significantly outperforms the state of the art (Wu et al., 2016) which instead uses

In Fig. 16 of the Appendix we include the confusion matrix of the classifier evaluated on our latent codes on ModelNet40 – the confusion happens between particularly similar geometries: a dresser vs. a nightstand or a flower-pot vs. a plant. The nuanced details that distinguish these objects may be hard to learn without stronger supervision.

	A	B	C	D	E	ours EMD	ours CD
MN10	79.8	79.9	-	80.5	91.0	**95.4**	**95.4**
MN40	68.2	75.5	74.4	75.5	83.3	84.0	**84.5**

Table 2. Classification performance (in %) on ModelNet10/40. Comparing to A: SPH (Kazhdan et al., 2003), B: LFD (Chen et al., 2003), C: T-L-Net (Girdhar et al., 2016), D: VConv-DAE (Sharma et al., 2016), E: 3D-GAN (Wu et al., 2016).

Figure 4. Point cloud *completions* of a network trained with partial and complete (input/output) point clouds and the EMD loss. Each triplet shows the partial input from the test split (left-most), followed by the network's output (middle) and the complete ground-truth (right-most).

Figure 5. Synthetic point clouds generated by samples produced with l-GAN (top) and 32-component GMM (bottom), both trained on the latent space of an AE using the EMD loss.

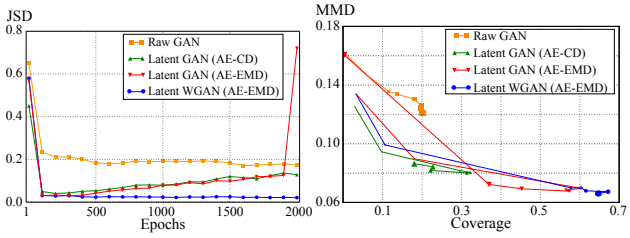


Figure 6. Learning behavior of the GANs, in terms of coverage / fidelity to the ground truth **test** dataset. Left – the JSD distance between the ground truth test set and synthetic datasets generated by the GANs at various epochs of training. Right – EMD based MMD/Coverage: curve markers indicate epochs 1, 10, 100, 200, 400, 1000, 1500, 2000, with *larger symbols denoting later epochs.*

5.2. Evaluating the generative models

Having established the quality of our AE, we now demonstrate the merits and shortcomings of our generative pipelines and establish one more successful application for the AE's learned representation. First, we conduct a comparison between our generative models followed by a comparison between our latent GMM generator and the state-of-the-art 3D voxel generator. Next, we describe how Chamfer distance can yield misleading results in certain pathological cases that r-GANs tends to produce. Finally, we show the benefit of working with a pre-trained latent representation in multi-class generators.

Comparison of our different generative models For this study, we train five generators with point clouds of the *chair* category. First, we establish two AEs trained with the CD or EMD loss respectively—referred to as AE-CD and AE-EMD and train an l-GAN in each latent space with the non-saturating loss of Goodfellow et al. (2014). In the space learned by the AE-EMD we train two additional models: an identical (architecture-wise) l-GAN that utilizes the Wasserstein objective with gradient-penalty (Gulrajani et al., 2017) and a family of GMMs with a different number of means and structures of covariances. We also train an r-GAN directly on the point cloud data.

Fig. 6 shows the JSD (left) and the MMD and Coverage (right) between the produced synthetic datasets and the held-out *test* data for the GAN-based models, as training proceeds. Note that the r-GAN struggles to provide good coverage and good fidelity of the test set; which alludes to the well-established fact that end-to-end GANs are generally difficult to train. The l-GAN (AE-CD) performs better in terms of fidelity with much less training, but its coverage remains low. Switching to an EMD-based AE for the representation and otherwise using the same latent GAN architecture (l-GAN, AE-EMD), yields a dramatic improvement in coverage and fidelity. Both l-GANs though suffer from the known issue of mode collapse: half-way through training, first coverage starts dropping with fidelity still at good levels, which implies that they are overfitting a small subset of the data. Later on, this is followed by a more catastrophic collapse, with coverage dropping as low as 0.5%. Switching to a latent WGAN largely eliminates this collapse, as expected.

In Table 3, we report measurements for all generators based on the epoch (or underlying GMM parameters) that has minimal JSD between the generated samples and the validation set. To reduce the sampling bias of these measurements each generator produces a set of synthetic samples that is $3\times$ the population of the comparative set (test or validation) and repeat the process 3 times and report the averages. The GMM selected by this process has 32 Gaussians and a full covariance. As shown in Fig. 18 of the Appendix, GMMs with full covariances perform much better than those that have diagonal structure and $\sim$20 Gaussians suffice for good

results. Last, the first row of Table 3 shows a baseline model that memorizes a random subset of the training data of the same size as the other generated sets.

Discussion. The results of Table 3 agree with the trends shown in Fig. 6 and further verify the superiority of the latent-based approaches and the relative gains of using an AE-EMD vs. an AE-CD. Moreover they demonstrate that a simple GMM can achieve results of comparable quality to a latent WGAN. Lastly, it is worth noting how the GMM has achieved similar fidelity as that of the perfect/memorized chairs and with almost as good coverage. Table 8 of the supplementary shows the same performance-based conclusions when our metrics are evaluated on the *train* split.

Model	Type	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
A	MEM	0.017	0.0018	0.063	78.6	79.4
B	RAW	0.176	0.0020	0.123	19.0	52.3
C	CD	0.048	0.0020	0.079	32.2	59.4
D	EMD	0.030	0.0023	0.069	57.1	59.3
E	EMD	0.022	0.0019	0.066	66.9	67.6
F	GMM	**0.020**	**0.0018**	**0.065**	**67.4**	**68.9**

Table 3. Evaluating 5 generators on the *test* split of the chair dataset on epochs/models selected via minimal JSD on the validation-split. We report: A: sampling-based memorization baseline, B: r-GAN, C: l-GAN (AE-CD), D: l-GAN (AE-EMD) , E: l-WGAN (AE-EMD), F: GMM (AE-EMD).

Chamfer's blindness, r-GAN's hedging. An interesting observation regarding r-GAN can be made in Table 3. The JSD and the EMD based metrics strongly favor the latent-approaches, while the Chamfer-based ones are much less discriminative. To decipher this discrepancy we did an extensive qualitative inspection of the r-GAN samples and found many cases of point clouds that were over-populated in locations, that on average, *most* chairs have mass. This hedging of the r-GAN is particularly hard for Chamfer to penalize since one of its two summands can become significantly small and the other can be only moderately big by the presence of a few sparsely placed points in the non-populated locations. Figure 7 highlights this point. For a ground-truth point cloud we retrieve its nearest neighbor, under the CD, in synthetically generated sets produced by the r-GAN and the l-GAN and in-image numbers report their CD and EMD distances from it. Notice how the CD fails to distinguish the inferiority of the r-GAN samples while the EMD establishes it. This blindness of the CD metric to only partially good matches, has the additional side-effect that the CD-based coverage is consistently bigger than the EMD-based one.

Comparisons to voxel generators. Generative models for other 3D modalities, like voxels, have been recently proposed (Wu et al., 2016). One interesting question is: if point clouds are our target modality, does it make sense to use voxel generators and then convert to point clouds? This experiment answers this question in the negative. First, we make a comparison using a latent GMM which is trained in conjunction with an AE-EMD. Secondly, we build an AE which operates with *voxels* and fit a GMM in the corresponding latent space. In both cases, we use 32 Gaussians and a full covariance matrix for these GMMs. To use our point-based metrics, we convert the output of (Wu et al., 2016) and our voxel-based GMM into meshes which we sample to generate point clouds. To do this conversion we use the marching-cubes (Lewiner et al., 2003) algorithm with an isovalue of 0.1 for the former method (per authors' suggestions) and 0.5 for our voxel-AE. We also constrain each mesh to be a single connected component as the vast majority of ground-truth data are.

Table 4 reveals how our point-based GMM trained with a class specific AE-EMD fares against (Wu et al., 2016) on four object classes for which the authors have made their (also class-specific) models publicly[2] available. Our

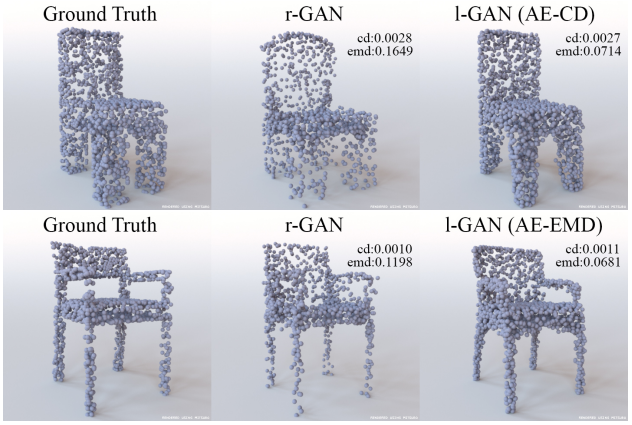


Figure 7. The CD distance is less faithful than EMD to visual quality of synthetic results; here, it favors r-GAN results, due to the overly high density of points in the seat part of the synthesized point sets.

Class	Fidelity		Coverage	
	A	Ours	A	Ours
car	0.059	**0.041**	28.6	**65.3**
rifle	0.051	**0.045**	69.0	**74.8**
sofa	0.077	**0.055**	52.5	**66.6**
table	0.103	**0.061**	18.3	**71.1**

Table 4. Fidelity (MMD-EMD) and coverage (COV-EMD) comparison between A: Wu et al. (2016) and our GMM generative model on the *test* split of each class. Note that Wu et al. uses *all* models of each class for training contrary to our generators.

[2] http://github.com/zck119/3dgan-release

Figure 8. Synthetic point clouds produced with l-WGANs trained in the latent space of an AE-EMD trained on a *multi-class* dataset.

approach is consistently better, with a coverage boost that can be as large as $4\times$ and an almost $2\times$ improved fidelity (case of table). This is despite the fact that (Wu et al., 2016) uses *a*ll models of each class for training, contrary to our generators that never had access to the underlying test split.

Table 5 reveals the performance achieved by pre-training a *voxel*-based AE for the chair class. Observe how by working with a voxel-based latent space, aside of making comparisons more direct to (Wu et al., 2016) (e.g. we both convert output voxels to meshes), we also establish significant gains in terms of coverage and fidelity.

	MMD-CD	MMD-EMD	COV-CD	COV-EMD
A	0.0046	0.091	19.6	22.4
Ours	**0.0025**	**0.072**	**60.3**	**64.8**

Table 5. MMD and Coverage metrics evaluated on the output of voxel-based methods at resolution 64^3, matched against the chair *test* set, using the same protocol as in Table3. Comparing: A: "raw" 64^3-voxel GAN (Wu et al., 2016) and a latent 64^3-voxel GMM.

Qualitative results In Fig. 5, we show some synthetic results produced by our l-GANs and the 32-component GMM. We notice high quality results from either model. The shapes corresponding to the 32 means of the Gaussian components can be found in the Appendix (Fig. 20), as well as results using the r-GAN (Fig.12).

Multi-class generators Finally, we compare between class specific and class agnostic generators. In Table 6 we report the MMD-CD for l-WGANs trained in the space of either a dedicated (per-class) AE-EMD or with an AE-EMD trained with all listed object classes. It turns out that the l-WGANs produce perform similar results in either space. Qualitative comparison (Fig. 8) also reveals that by using a multi-class AE-EMD we do not sacrifice much in terms of visual quality compared to the dedicated AEs.

6. Related Work

Recently, deep learning architectures for view-based projections (Su et al., 2015; Wei et al., 2016; Kalogerakis et al., 2016), volumetric grids (Qi et al., 2016b; Wu et al., 2015; Hegde & Zadeh, 2016) and graphs (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Yi et al., 2016b) have appeared in the 3D machine learning literature.

A few recent works ((Wu et al., 2016), (Wang et al., 2016),

	airplane	*car*	*chair*	*sofa*	*table*	average	multi-class
Tr	0.0004	0.0006	0.0015	0.0011	0.0013	**0.0010**	0.0011
Te	0.0006	0.0007	0.0019	0.0014	0.0017	**0.0013**	0.0014

Table 6. MMD-CD measurements for l-WGANs trained on the latent spaces of dedicated (left 5 columns) and multi-class EMD-AEs (right column). Also shown is the weighted average of the per-class values, using the number of train (Tr) resp. test (Te) examples of each class as weights. All l-WGANs use the model parameter resulted by 2000 epochs of training.

(Girdhar et al., 2016), (Brock et al., 2016), (Maimaitimin et al., 2017), (Zhu et al., 2016)) have explored generative and discriminative representations for geometry. They operate on different modalities, typically voxel grids or view-based image projections. To the best of our knowledge, our work is the first to study such representations for point clouds.

Training Gaussian mixture models (GMM) in the latent space of an autoencoder is closely related to VAEs (Kingma & Welling, 2013). One documented issue with VAEs is over-regularization: the regularization term associated with the prior, is often so strong that reconstruction quality suffers (Bowman et al., 2015; Sønderby et al., 2016; Kingma et al., 2016; Dilokthanakul et al., 2016). The literature contains methods that start only with a reconstruction penalty and slowly increase the weight of the regularizer. An alternative approach is based on adversarial autoencoders (Makhzani et al., 2015) which use a GAN to implicitly regularize the latent space of an AE.

7. Conclusion

We presented a novel set of architectures for 3D point cloud representation learning and generation. Our results show good generalization to unseen data and our representations encode meaningful semantics. In particular our generative models are able to produce faithful samples and cover most of the ground truth distribution. Interestingly, our extensive experiments show that the best generative model for point clouds is a GMM trained in the fixed latent space of an AE. While this might not be a universal result, it suggests that simple classic tools should not be dismissed. A thorough investigation on the conditions under which simple latent GMMs are as powerful as adversarially trained models would be of significant interest.

Acknowledgements

The authors wish to thank all the anonymous reviewers for their insightful comments and suggestions. Lin Shao and Matthias Nießner for their help with the shape-completions and Fei Xia for his suggestions on the evaluation metrics. Last but not least, they wish to acknowledge the support of NSF grants NSF IIS-1528025 and DMS-1546206, ONR MURI grant N00014-13-1-0341, a Google Focused Research award, and a gift from Amazon Web Services for Machine Learning Research.

References

Arora, S. and Zhang, Y. Do gans actually learn the distribution? an empirical study. *CoRR*, abs/1706.08224, 2017.

Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. Dynamic FAUST: Registering human bodies in motion. In *IEEE CVPR*, 2017.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. Mode regularized generative adversarial networks. *CoRR*, abs/1612.02136, 2016.

Chen, D.-Y., Tian, X.-P., Shen, Y.-T., and Ouhyoung, M. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, 2003.

Dai, A., Qi, C. R., and Nießner, M. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *http://arxiv.org/abs/1612.00101*, 2016.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1), 1977.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR*, abs/1611.02648, 2016.

Fan, H., Su, H., and Guibas, L. J. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016.

Girdhar, R., Fouhey, D. F., Rodriguez, M., and Gupta, A. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

Hegde, V. and Zadeh, R. Fusionnet: 3d object classification using multiple data representations. *CoRR*, abs/1607.05695, 2016.

Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

Huang, R., Achlioptas, P., Guibas, L., and Ovsjanikov, M. Latent space representation for shape analysis and learning. *http://arxiv.org/abs/1806.03967*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jakob, W. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. 3d shape segmentation with projective convolutional networks. *CoRR*, abs/1612.02808, 2016.

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *ACM SGP*, 2003.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kingma, D. P., Salimans, T., and Welling, M. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016.

Kullback, S. and Leibler, R. A. On information and sufficiency. *Annals of Mathematical Statistics*, 1951.

Lewiner, T., Lopes, H., Vieira, A. W., and Tavares, G. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 2003.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Maimaitimin, M., Watanabe, K., and Maeyama, S. Stacked convolutional auto-encoders for surface recognition based on 3d point cloud data. *Artificial Life and Robotics*, 2017.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016a.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. Volumetric and multi-view cnns for object classification on 3d data. In *IEEE CVPR*, 2016b.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, 2017.

Radford, A., Metz, L., and Chintala, S. *CoRR*, abs/1511.06434.

Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. *IJCV*, 2000.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

Rustamov, R. M., Ovsjanikov, M., Azencot, O., Ben-Chen, M., Chazal, F., and Guibas, L. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph.*, 32(4), July 2013.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *NIPS*, 2016.

Sharma, A., Grau, O., and Fritz, M. Vconv-dae: Deep volumetric shape learning without object labels. In *ECCV Workshop*, 2016.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. *CoRR*, abs/1602.02282, 2016.

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

Sung, M., Kim, V. G., Angst, R., and Guibas, L. J. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 34(6):175, 2015.

Tasse, F. P. and Dodgson, N. Shape2vec: Semantic-based descriptors for 3d shapes, sketches and images. *ACM Trans. Graph.*, 2016.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *CoRR*, abs/1703.09438, 2017.

Wang, Y., Xie, Z., Xu, K., Dou, Y., and Lei, Y. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. *Neurocomputing*, 174, 2016.

Wei, L., Huang, Q., Ceylan, D., Vouga, E., and Li, H. Dense human body correspondences using convolutional networks. In *IEEE CVPR*, 2016.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *NIPS*. 2016.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *IEEE CVPR*, 2015.

Yi, L., Kim, V. G., Ceylan, D., Shen, I., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. J. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6), 2016a.

Yi, L., Su, H., Guo, X., and Guibas, L. J. Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation. *CoRR*, abs/1612.00606, 2016b.

Zhu, Z., Wang, X., Bai, S., Yao, C., and Bai, X. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 2016.

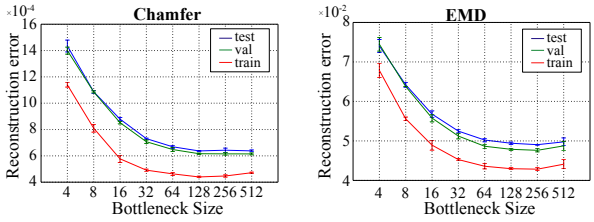


Figure 9. The bottleneck size was fixed at 128 in all single-class experiments by observing the reconstruction loss of the AEs, shown here for various bottleneck sizes, when training with the data of the chair class.

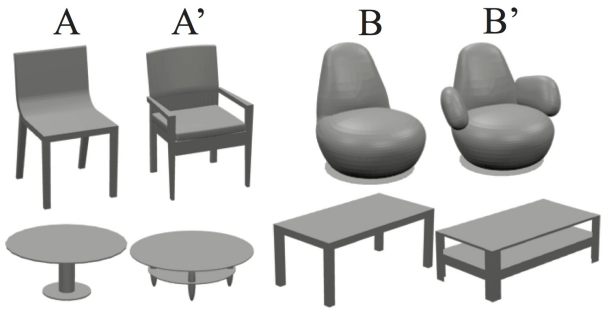


Figure 10. Shape Analogies using our learned representation. Shape B' relates to B in the same way that shape A' relates to A.

A. AE Details

The encoding layers of our AEs were implemented as 1D-convolutions with ReLUs, with kernel size of 1 and stride of 1, i.e. treating each 3D point independently. Their decoding layers, were MLPs built with FC-ReLUs. We used Adam (Kingma & Ba, 2014) with initial learning rate of 0.0005, β_1 of 0.9 and a batch size of 50 to train all AEs.

A.1. AE used for SVM-based experiments

For the AE mentioned in the SVM-related experiments of Section 5.1 of the main paper, we used an encoder with $128, 128, 256$ and 512 filters in each of its layers and a decoder with $1024, 2048, 2048 \times 3$ neurons, respectively. Batch normalization was used between every layer. We also used online data augmentation by applying random rotations along the gravity-(z)-axis to the input point clouds of each batch. We trained this AE for 1000 epochs with the CD loss and for 1100 with the EMD.

A.2. All other AEs

For all other AEs, the encoder had $64, 128, 128, 256$ and k filters at each layer, with k being the bottle-neck size. The decoder was comprised by 3 FC-ReLU layers with $256, 256, 2048 \times 3$ neurons each. We trained these AEs for a maximum of 500 epochs when using single class data and 1000 epochs for the experiment involving 5 shape classes (end of Section 5.2, main paper).

A.3. AE regularization

To determine an appropriate size for the latent-space, we constructed 8 (otherwise architecturally identical) AEs with bottleneck sizes $k \in \{4, 8 \ldots, 512\}$ and trained them with point clouds of the chair object class, under the two losses (Fig. 9). We repeated this procedure with pseudo-random weight initializations three times and found that $k = 128$ had the best generalization error on the test data, while achieving minimal reconstruction error on the train split.

Remark. Different AE setups brought no noticeable advantage over our main architecture. Concretely, adding drop-out layers resulted in worse reconstructions and using batch-norm on the encoder *only*, sped up training and gave us slightly better generalization error when the AE was trained with single-class data. Exclusively, for the SVM experiment of Section 5.1 of the main paper we randomly rotate the input chairs to promote latent features that are rotation-invariant.

B. Applications of the Latent Space Representation

For shape editing applications, we use the embedding we learned with the AE-EMD trained *across all* 55 object classes, not separately per-category. This showcases its ability to encode features for different shapes, and enables interesting applications involving different kinds of shapes.

Editing shape parts. We use the shape annotations of Yi et al.(Yi et al., 2016a) as guidance to modify shapes. As an example, assume that a given object category (e.g. chairs) can be further subdivided into two sub-categories $\mathcal{A}$ and $\mathcal{B}$: every object $A \in \mathcal{A}$ possesses a certain structural property (e.g. has armrests, is four-legged, etc.) and objects $B \in \mathcal{B}$ do not. Using our latent representation we can model this structural difference between the two sub-categories by the difference between their average latent representations $\mathbf{x}_\mathcal{B} - \mathbf{x}_\mathcal{A}$, where $\mathbf{x}_\mathcal{A} = \sum_{A \in \mathcal{A}} \mathbf{x}_A$, $\mathbf{x}_\mathcal{B} = \sum_{B \in \mathcal{B}} \mathbf{x}_B$. Then, given an object $A \in \mathcal{A}$, we can change its property by transforming its latent representation: $x_{A'} = x_A + \mathbf{x}_\mathcal{B} - \mathbf{x}_\mathcal{A}$, and decode $\mathbf{x}_{A'}$ to obtain $A' \in \mathcal{B}$. This process is shown in Fig. 3 of the main paper.

Interpolating shapes. By linearly interpolating between the latent representations of two shapes and decoding the result we obtain intermediate variants between the two shapes. This produces a "morph-like" sequence with the two shapes at its end points Fig. 2 of main paper and Fig. 11 here). Our latent representation is powerful enough to support re-

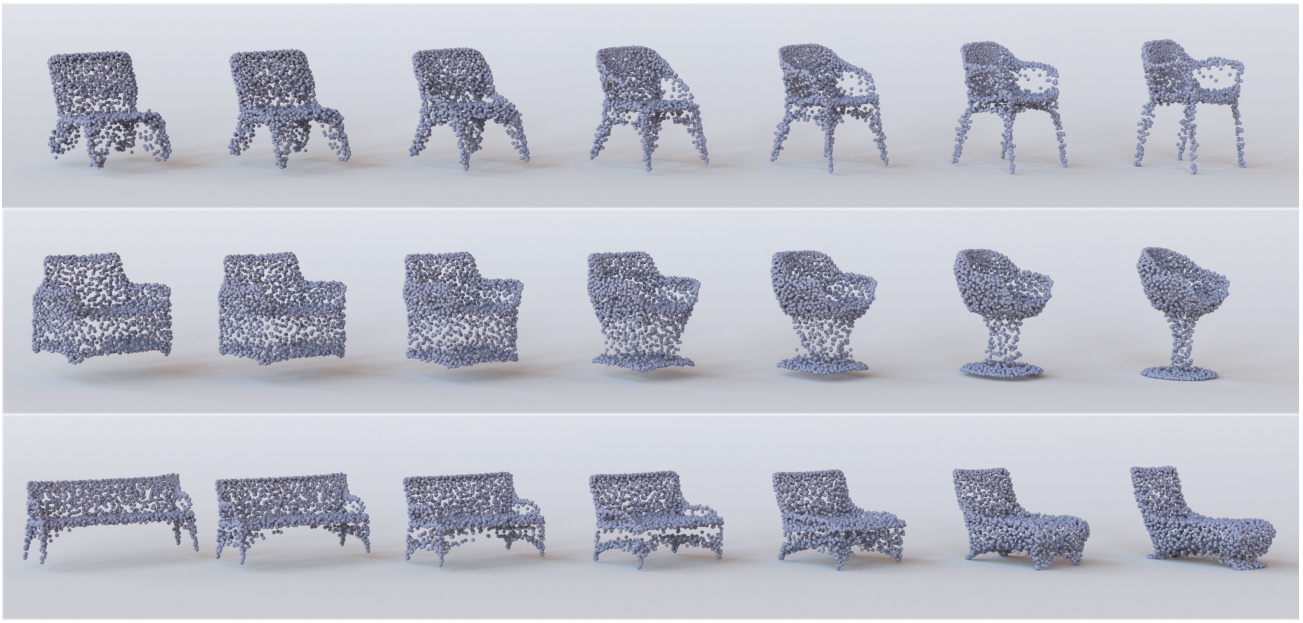


Figure 11. Interpolating between different point clouds (left and right-most of each row), using our latent space representation. Note the interpolation between structurally and topologically different shapes. **Note**: for all our illustrations that portray point clouds we use the Mitsuba renderer (Jakob, 2010).

moving and merging shape parts, which enables morphing between shapes of significantly different appearance. Our cross-category latent representation enables morphing between shapes of different classes, cfg. the second row for an interpolation between a bench and a sofa.

Shape analogies. Another demonstration of the Euclidean nature of the latent space is demonstrated by finding analogous shapes by a combination of linear manipulations and Euclidean nearest-neighbor searching. Concretely, we find the difference vector between A and A', we add it to shape B and search in the latent space for the nearest-neighbor of that result, which yields shape B'. We demonstrate the finding in Fig. 10 with images taken from the meshes used to derive the underlying point clouds to help the visualization. Finding shape analogies has been of interest recently in the geometry processing community (Rustamov et al., 2013; Huang et al., 2018).

Loss	ModelNet40			ModelNet10		
	C-plt	icpt	loss	C-plt	icpt	loss
EMD	0.09	0.5	hng	0.02	3	sq-hng
CD	0.25	0.4	sq-hng	0.05	0.2	sq-hng

Table 7. Training parameters of SVMs used in each dataset with each structural loss of the AE. *C-penalty (C-plt)*: term controlling the trade-off between the size of the learned margin and the misclassification rate; *intercept (icpt)*: extra dimension appended on the input features to center them; *loss*: svm's optimization loss function: hinge (*hng*), or squared-hinge (*sq-hng*).

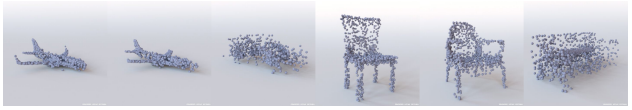


Figure 12. Synthetic results produced by the r-GAN. From left to right: airplanes, car, chairs, sofa.

C. Autoencoding Human Forms

In addition to ShapeNet core which contains man-made only objects, we have experimented with the D-FAUST dataset of (Bogo et al., 2017) that contains meshes of human subjects. Specifically, D-FAUST contains 40K scanned meshes of 10 human subjects performing a variety of motions. Each human performs a set of (maximally) 14 motions, each captured by a temporal sequence of ~300 meshes. For our purposes, we use a random subset of 80 (out of the 300) meshes for each human/motion and extract from each mesh a point cloud with 4096 points. Our resulting dataset contains a total of 10240 point clouds and we use a train-test-val split of $[70\%, 20\%, 10\%]$ - while enforcing that every split contains *all* human/motion combinations. We use this data to train and evaluate an AE-EMD that is identical to the single-class AE presented in the main paper, with the only difference being the number of neurons of the last layer (4096×3 instead of 2048×3).

We demonstrate reconstruction and interpolation results in Figs. 13 and 14. For a given human subject and a specific motion we pick at random two meshes corresponding to time points t_0, t_1 (with $t_1 > t_0$) and show their reconstruc-

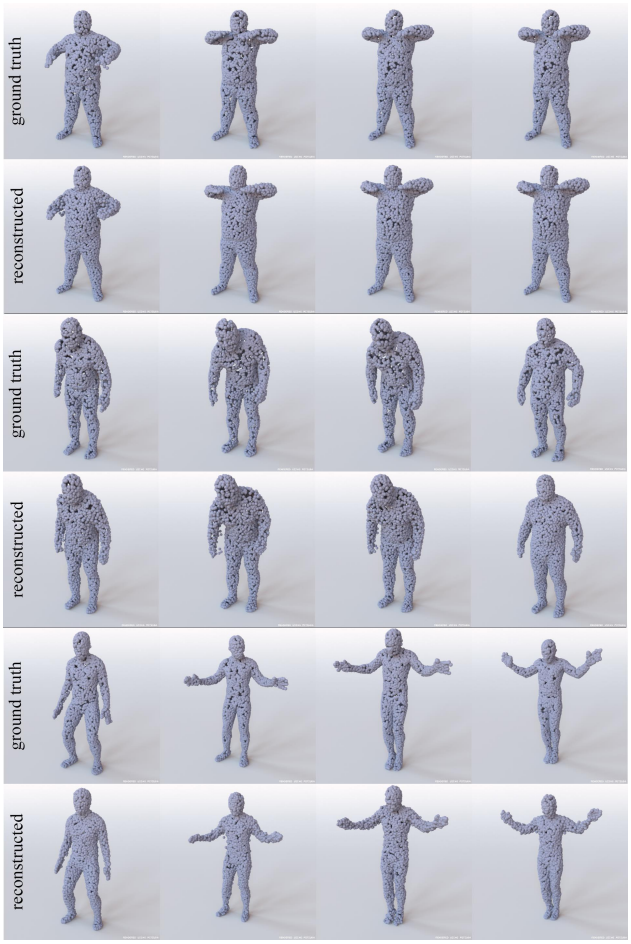


Figure 13. Reconstructions of unseen shapes from the *test* split extracted from the D-FAUST dataset of (Bogo et al., 2017) with an AE-EMD decoding point clouds with 4096 points. In each row the poses depict a motion (left-to-right) as it progress in time.

tions along with the ground truth in Fig. 13 (left-most and right-most of each row). In the same figure we also plot the reconstructions of two random meshes captured in (t_0, t_1) (middle-two of each row). In Fig. 14, instead of encoding/decoding the ground truth test data, we show decoded linear *interpolations* between the meshes of t_0, t_1.

D. Shape Completions

An important application that our AE architecture can be used for is that of completing point clouds that contain limited information of the underlying geometry. Typical range scans acquired for an object in real-time can often miss entire regions of the object due to the existence of self-occlusions and the lack of adequate (or "dense") view-point registrations. This fact makes any sensible solution to this problem of high practical importance. To address it here, we resort in a significantly different dataset than the ones used in the rest of this paper. Namely, we utilize the dataset of (Dai et al., 2016) that contains pairs of complete (intact) 3D CAD models and *partial* versions of them. Specifically, for each object of ShapeNet (core) it contains six partial point clouds created by the aggregation of frames taken over a limited set of view-points in a virtual trajectory established around the object. Given this data, we first fix the dimensionality of the partial point clouds to be 2048 points for each one by randomly sub-sampling them. Second, we apply uniform-in-area sampling to each complete CAD model to extract from it 4096 points to represent a "complete" ground-truth datum. All the resulting point clouds are centered in the unit-sphere and (within a class) the partial and complete point clouds are co-aligned. Last, we train class-specific neural-nets with Chair, Table and Airplane data and a train/val/test split of [80%, 5%, 15%].

D.1. Architecture

The high level design of the architecture we use for shape-completions is identical to the AE, i.e. independent-convolutions followed by FCs, trained under a structural loss (CD or EMD). However, essential parts of this network are different: depth, bottleneck size (controlling compression ratio) and the crucial differentiation between the input and the output data. Technically, the resulting architecture is an Abstractor-Predictor (AP) and is comprised by three layers of independent per-point convolutions, with filter sizes of $[64, 128, 1024]$, followed by a max-pool, which is followed by an FC-ReLU (1024 neurons) and a final FC layer (4096×3 neurons). We don't use batch-normalization between any layer and train each class-specific AP for a maximum of 100 epochs, with ADAM, initial learning rate of 0.0005 and a batch size of 50. We use the minimal per the validation split model (epoch) for evaluating our models with the test data.

D.2. Evaluation

We use the specialized point cloud completion metrics introduced in (Sung et al., 2015). That is a) the *accuracy*: which is the fraction of the predicted points that are within a given radius (ρ) from any point in the ground truth point cloud and b) the *coverage*: which is the fraction of the ground-truth points that are within ρ from any predicted point. In Table 8 we report these metrics (with a $\rho = 0.02$ similarly to (Sung et al., 2015)) for class-specific networks that were trained with the EMD and CD losses respectively. We observe that the CD loss gives rise to more accurate but also less complete outputs, compared to the EMD. This highlights again the greedy nature of CD – since it does not take into account the matching between input/output, it can get generate completions that are more concentrated around the (incomplete) input point cloud. Figure 15 shows the corresponding completions of those presented in the main paper, but with a network trained under the CD loss.

Figure 14. Interpolating between different point clouds from the *test* split (left and right-most of each row) of the D-FAUST dataset of (Bogo et al., 2017). These linear interpolations have captured some of the dynamics of the corresponding motions: 'chicken-wings' (first row), 'shake shoulders' (second row) and 'jumping jacks' (third row). Compare to Fig.13 that contains ground-truth point clouds in the same time interval.

Figure 15. Point cloud *completions* of a network trained with partial and complete (input/output) point clouds and the **CD loss**. Each triplet shows the partial input from the test split (left-most), followed by the network's output (middle) and the complete ground-truth (right-most). Also compare with Fig. 4 of main paper that portrays the corresponding completions of a network trained with the EMD loss.

Class	Airplane	Chair	Table
Test-size	4.5K	6K	6K
Acc-CD	**96.9**	**86.5**	**87.6**
Acc-EMD	94.7	77.1	78.4
Cov-CD	96.6	77.5	75.2
Cov-EMD	**96.8**	**82.6**	**83.0**

Table 8. Performance of point cloud *completions* on ShapeNet *test* data. Comparison between Abstractor-Predictors trained under the CD or EMD losses, on mean **Acc**uracy and **Cov**erage, across each class. The size of each test-split is depicted in the first row.

E. SVM Parameters for Auto-encoder Evaluation

For the classification experiments of Section 5.1 (main paper) we used a one-versus-rest linear SVM classifier with an l_2 norm penalty and balanced class weights. The exact optimization parameters can be found in Table 7. The confusion matrix of the classifier evaluated on our latent codes on ModelNet40 is shown in Fig. 16.

F. r-GAN Details

The discriminator's first 5 layers are 1D-convolutions with stride/kernel of size 1 and $\{64, 128, 256, 256, 512\}$ filters each; interleaved with leaky-ReLU. They are followed by a feature-wise max-pool. The last 2 FC-leaky-ReLU layers have $\{128, 64\}$, neurons each and they lead to single sigmoid neuron. We used 0.2 units of leak.

The generator consists of 5 FC-ReLU layers with $\{64, 128, 512, 1024, 2048 \times 3\}$ neurons each. We trained r-GAN with Adam with an initial learning rate of 0.0001, and $beta_1$ of 0.5 in batches of size 50. The noise vector was drawn by a spherical Gaussian of 128 dimensions with zero mean and 0.2 units of standard deviation.

Some synthetic results produced by the r-GAN are shown in Fig. 12.

G. l-GAN Details

The discriminator consists of 2 FC-ReLU layers with $\{256, 512\}$ neurons each and a final FC layer with a single sigmoid neuron. The generator consists of 2 FC-ReLUs with

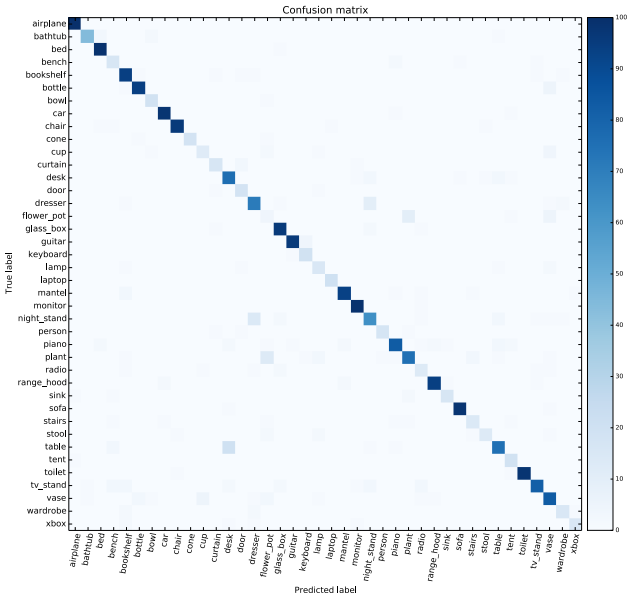


Figure 16. Confusion matrix for the SVM-based classification of Section 5.1, for the Chamfer loss on ModelNet40. The class pairs most confused by the classifier are dresser/nightstand, flower pot/plant. Better viewed in the electronic version.

$\{128, k = 128\}$ neurons each. When used the l-Wasserstein-GAN, we used a gradient penalty regularizer $\lambda = 10$ and trained the critic for 5 iterations for each training iteration of the generator. The training parameters (learning rate, batch size) and the generator's noise distribution were the same as those used for the r-GAN.

H. Model Selection of GANs

All GANs are trained for maximally 2000 epochs; for each GAN, we select one of its training epochs to obtain the "final" model, based on how well the synthetic results match the ground-truth distribution. Specifically, at a given epoch, we use the GAN to generate a set of synthetic point clouds, and measure the distance between this set and the validation set. We avoid measuring this distance using MMD-EMD, given the high computational cost of EMD. Instead, we use either the JSD or MMD-CD metrics to compare the synthetic dataset to the validation dataset. To further reduce the computational cost of model selection, we only check every 100 epochs (50 for r-GAN). The generalization error of the various GAN models, at various training epochs, as measured by MMD and JSD is shown in Fig. 17 (left and middle).

Using the same JSD criterion, we also select the number and covariance type of Gaussian components for the GMM (Fig. 18, left), and obtain the optimal value of 32 components. GMMs performed much better with full (as opposed to diagonal) covariance matrices, suggesting strong correlations between the latent dimensions (Fig. 18, right).

When using MMD-CD as the selection criterion, we obtain models of similar quality and at similar stopping epochs (see Table 9); the optimal number of Gaussians in this case was 40. The training behavior measured using MMD-CD can be seen in Fig. 17 (right).

Method	Epoch	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
A	1350	0.1893	0.0020	0.1265	19.4	54.7
B	300	0.0463	0.0020	0.0800	32.6	58.2
C	200	0.0319	0.0022	0.0684	57.6	58.7
D	1700	0.0240	0.0020	0.0664	64.2	64.7
E	-	**0.0182**	**0.0018**	**0.0646**	**68.6**	**69.3**

Table 9. Evaluation of five generators on **test-split** of *chair* data on epochs/models that were selected via minimal MMD-CD on the validation-split. We report: A: r-GAN, B: l-GAN (AE-CD), C: l-GAN (AE-EMD) , D: l-WGAN (AE-EMD), E: GMM-40-F (AE-EMD). GMM-40-F stands for a GMM with 40 Gaussian components with full covariances. The reported scores are averages of three pseudo-random repetitions. Compare this with Table 3 of the main paper. Note that the overall quality of the selected models remains the same, irrespective of the metric used for model selection.

I. Voxel AE Details

Our voxel-based AEs are fully-convolutional with the encoders consisting of 3D-Conv-ReLU layers and the decoders of 3D-Conv-ReLU-transpose layers. Below, we list the parameters of consecutive layers, listed left-to-right. The layer parameters are denoted in the following manner: (number of filters, filter size). Each Conv/Conv-Transpose has a stride of 2 except the last layer of the 32^3 decoder which has 4. In the last layer of the decoders we do not use a non-linearity. The abbreviation "bn" stands for batch-normalization.

- 32^3 - model
  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 2) $\rightarrow$ bn $\rightarrow$ (64, 2)
  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (1, 8) $\rightarrow$ Output

- 64^3 - model
  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 4) $\rightarrow$ bn $\rightarrow$ (64, 2) $\rightarrow$ (64, 2)
  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (32, 8) $\rightarrow$ (1, 8) $\rightarrow$ Output

We train each AE for 100 epochs with Adam under the binary cross-entropy loss. The learning rate was 0.001, the

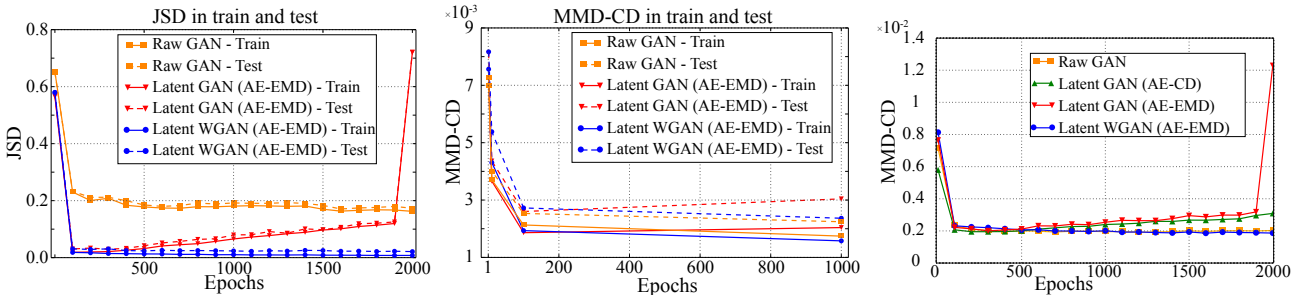


Figure 17. Left/middle: Generalization error of the various GAN models, at various training epochs. Generalization is estimated using the JSD (left) and MMD-CD (middle) metrics, which measure closeness of the synthetic results to the training resp. test ground truth distributions. The plots show the measurements of various GANs. Right: Training trends in terms of the MMD-CD metric for the various GANs. Here, we sample a set of synthetic point clouds for each model, of size 3x the size of the ground truth test dataset, and measure how well this synthetic dataset matches the ground truth in terms of MMD-CD. This plot complements Fig. 6 (left) of the main paper, where a different evaluation measure was used - note the similar behavior.

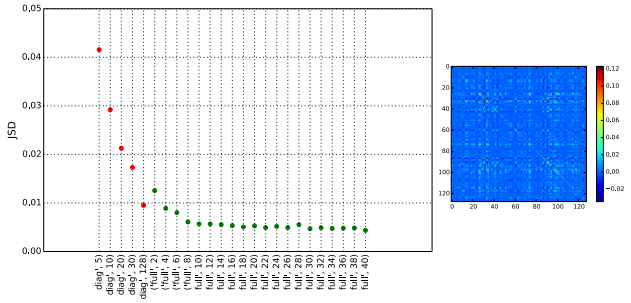


Figure 18. GMM model selection. GMMs with a varying number of Gaussians and covariance type are trained on the latent space learned by and AE trained with EMD and a bottleneck of 128. Models with a full covariance matrix achieve significantly smaller JSD than models trained with diagonal covariance. For those with full covariance, 30 or more clusters seem sufficient to achieve minimal JSD. On the right, the values in a typical covariance matrix of a Gaussian component are shown in pseudocolor - note the strong off-diagonal components.

β_1 0.9 and the batch size 64. To validate our voxel AE architectures, we compared them in terms of reconstruction quality to the state-of-the-art method of (Tatarchenko et al., 2017) and obtained comparable results, as demonstrated in Table 10.

Voxel Resolution	32	64
Ours	92.7	88.4
(Tatarchenko et al., 2017)	93.9	90.4

Table 10. Reconstruction quality statistics for our dense voxel-based AE and the one of (Tatarchenko et al., 2017) for the ShapeNetCars dataset. Both approaches use a 0.5 occupancy threshold and the train-test split of (Tatarchenko et al., 2017). Reconstruction quality is measured by measuring the intersection-over-union between the input and synthesized voxel grids, namely the ratio between the volume in the voxel grid that is 1 in both grids divided by the volume that is 1 in at least one grid.

J. Memorization Baseline

Here we compare our GMM-generator against a model that memorizes the training data of the chair class. To do this, we either consider the entire training set or randomly subsample it, to create sets of different sizes. We then evaluate our metrics between these memorized sets and the point clouds of test split (see Table 11). The coverage/fidelity obtained by our generative models (last row) is slightly lower than the equivalent in size case (third row) as expected: memorizing the training set produces good coverage/fidelity with respect to the test set when they are both drawn from the same population. This speaks for the validity of our metrics. Naturally, the advantage of using a learned representation lies in learning the structure of the underlying space instead of individual samples, which enables compactly representing the data and generating novel shapes as demonstrated by our interpolations. In particular, note that while some mode collapse is present in our generative results, as indicated by the $\sim$10% drop in coverage, the MMD of our generative models is almost identical to that of the memorization case, indicating excellent fidelity.

K. More Comparisons with Wu et al.

In addition to the EMD-based comparisons in Table 4 of the main paper, in Tables 12, 13 we provide comparisons with (Wu et al., 2015) for the ShapeNet classes for which the authors have made publicly available their models. In Table 12 we provide JSD-based comparisons for two of our models. In Table 13 we provide Chamfer-based Fidelity/Coverage comparisons on the *test* split, that complement the EMD-based ones of Table 4 in the main paper.

Sample Set Size	COV-CD	MMD-CD	COV-EMD	MMD-EMD
Entire —Train—	97.3	0.0013	98.2	0.0545
1 × —Test—	54.0	0.0023	51.9	0.0699
3 × —Test—	79.4	0.0018	78.6	0.0633
Full-GMM/32 (3 × —Test—)	68.9	0.0018	67.4	0.0651

Table 11. Quantitative results of a baseline sampling/memorizing model, for different sizes of sets sampled from the training set and evaluated against the test split. The first three rows show results of a memorizing model, while the third row corresponds to our generative model. The first row shows the results of memorizing the entire training chair dataset. The second and third rows show the averages of three repetitions of the sub-sampling procedure with different random seeds.

Class	A	B		C	
	Tr+Te	Tr	Te	Tr	Te
airplane	-	0.0149	0.0268	**0.0065**	0.0191
car	0.1890	0.0081	0.0109	**0.0063**	0.0108
rifle	0.2012	0.0212	0.0364	**0.0092**	0.0214
sofa	0.1812	**0.0102**	0.0102	**0.0102**	0.0101
table	0.2472	0.0058	0.0177	**0.0035**	0.0143

Table 12. JSD-based comparison between A: (Wu et al., 2016) and our generative models – B: a latent GAN, C: our GM with 32 full-covariance Gaussian components. Both B and C were trained on the latent space of our AE with the EMD structural loss. Note that the l-GAN here uses the same "vanilla" adversarial objective as (Wu et al., 2016). Tr: train split, Te: test split.

Comparisons on training data. In Table 14 we compare to (Wu et al., 2016) in terms of the JSD and MMD-CD on the training set of the *chair* category. Since (Wu et al., 2016) do not use any train/test split, we perform 5 rounds of sampling 1K synthetic results from their models and report the best values of the respective evaluation metrics. We also report the average classification probability of the synthetic samples to be classified as chairs by the PointNet classifier. The r-GAN mildly outperforms (Wu et al., 2016) in terms of its diversity (as measured by JSD/MMD), while also creating realistic-looking results, as shown by the classification score. The l-GANs perform even better, both in terms of classification and diversity, with less training epochs. Finally, note that the PointNet classifier was trained on ModelNet, and (Wu et al., 2016) occasionally generates shapes that only rarely appear in ModelNet. In conjunction with their higher tendency for mode collapse, this partially accounts for their lower classification scores.

Class	MMD-CD		COV-CD	
	A	B	A	B
airplane	-	0.0005	-	71.1
car	0.0015	**0.0007**	22.9	**63.0**
rifle	0.0008	**0.0005**	56.7	**71.7**
sofa	0.0027	**0.0013**	42.40	**75.5**
table	0.0058	**0.0016**	16.7	**71.7**

Table 13. CD based MMD and Coverage comparison between A: Wu et al. (2016) and B: our generative model on the *test* split of each class. Our generative model is a GM with 32 full-covariance Gaussian components, trained on the latent space of our AE with the EMD structural loss. Note that Wu et al. used *a*ll models of each class for training.

Metric	A	B	C	D	E	F
JSD	0.1660	0.1705	0.0372	0.0188	0.0077	**0.0048**
MMD-CD	0.0017	0.0042	0.0015	0.0018	0.0015	**0.0014**
CLF	84.10	87.00	96.10	94.53	89.35	87.40

Table 14. Evaluating six generators on **train-split** of *chair* dataset on epochs/models selected via minimal JSD on the validation-split. We report: A: r-GAN, B: (Wu et al., 2016) (a volumetric approach), C: l-GAN(AE-CD), D: l-GAN(AE-EMD), E: l-WGAN(AE-EMD), F: GMM(AE-EMD). Note that the average classification score attained by the ground-truth point clouds was 84.7%.

L. Limitations

Figure 19 shows some failure cases of our models. Chairs with rare geometries (left two images) are sometimes not faithfully decoded. Additionally, the AEs may miss high-frequency geometric details, e.g. a hole in the back of a chair (middle), thus altering the style of the input shape. Finally, the r-GAN often struggles to create realistic-looking shapes (right) – while the r-GAN chairs are easily visually recognizable, it has a harder time on cars. Designing more robust raw-GANs for point clouds remain an interesting avenue for future work. A limitation of our shape-completion pipeline regards the style of the partial shape, which might not be well preserved in the completed point cloud (see Fig. 21 for an example).

Figure 19. The AEs might fail to reconstruct uncommon geometries or might miss high-frequency details: first four images - left of each pair is the input, right the reconstruction. The r-GAN may synthesize noisy/unrealistic results, cf. a car (right most image).

Figure 20. The 32 centers of the GMM fitted to the latent codes, and decoded using the decoder of the AE-EMD.

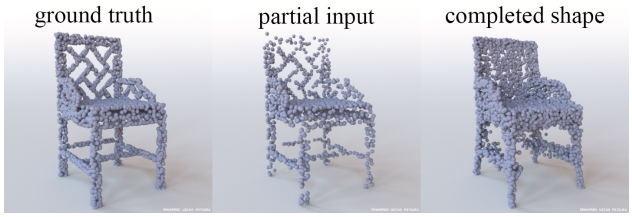


Figure 21. Our completion network might fail to preserve some of the style information in the partial point cloud, even though a reasonable shape is produced.